UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
INSTITUTO FEDERAL DE EDUCAÇÃO,
CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO - PROPG
MESTRADO PROFISSIONAL EM COMPUTAÇÃO APLICADA

PABLO DIEGO ALENCAR CARDOSO

COISA: CONSELHEIRO INTELIGENTE DE SAÚDE DO PROJETO LARIISA

FORTALEZA
2015

PABLO DIEGO ALENCAR CARDOSO

COISA: CONSELHEIRO INTELIGENTE DE SAÚDE DO PROJETO LARIISA

Dissertação submetida à Coordenação do Curso de Mestrado Profissional em Computação Aplicada da Universidade Estadual do Ceará e do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Antônio Mauro Barbosa de Oliveira.

FORTALEZA
2015

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Biblioteca Central Prof. Antônio Martins Filho

Cardoso, Pablo Diego Alencar.
COISA: Conselheiro Inteligente de Saúde do Projeto LARIISA [recurso eletrônico] / Pablo Diego Alencar Cardoso. – 2015.
1 CD-ROM: il.; 4 ¾ pol.

CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico com 91 folhas, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm).

Dissertação (mestrado profissional) – Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Mestrado Profissional em Computação Aplicada, Fortaleza, 2015.
Área de concentração: Redes de Computadores.
Orientação: Prof. Ph.D. Antônio Mauro Barbosa de Oliveira.
Coorientação: Prof. Dr. César Olavo de Moura Filho.

1. saúde. 2. monitoramento e alertas de doenças. 3. openEHR. 4. ontologia. 5. LARIISA. I. Título.

Pablo Diego Alencar Cardoso

COISA: CONSELHEIRO INTELIGENTE DE SAÚDE DO PROJETO LARIISA

Dissertação apresentada ao Curso de Mestrado Profissional em Computação Aplicada da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestrado em Computação.

Defesa em: **13/07/2015**

BANCA EXAMINADORA

Antonio Mauro Barbosa de Oliveira, PD. DSc. (IFCE)
Presidente (Orientador)

Prof. Dr. Luiz Odorico Monteiro de Andrade, DSc. (UFC)
Membro Externo

Prof. Dr. Ronaldo Fernandes Ramos, DSc. (IFCE)
Membro Interno

Dedico este trabalho aos meus progenitores.

AGRADECIMENTOS

Agradeço a todos que estiveram comigo ao longo desse caminho da ciência, percorrido com tantos sacrifícios. Também à Força que me motiva para continuar e persistir em meus objetivos.

Em especial, agradeço aos meus pais, Maria Helena Alencar e Francisco Pereira Cardoso, que me forneceram boa base para a escrita desse trabalho, e para manter-me na luta a fim de concluir meus objetivos. Aos meus irmãos, Paolo, Candice e Priscilla, que também serviram de espelho para minha determinação. Agradeço ainda à minha namorada Priscylla, pela paciência e tolerância diante da minha falta de tempo em determinadas datas, além da força que me deu nos momentos de estresse e correria.

Um abraço a meus amigos Ricardo Brasileiro, Valbert Wendel e Rafael de Brito, por sempre estarem dispostos a me escutar, além da motivação que a mim dispenderam enquanto eu estudava para ser Mestre. Agradeço a meus colegas de classe Evandro Soares, Odara, Reivel e Emanuel Dantas, que figuraram como grandes parceiros durante a caminhada. Também a toda equipe do MPCOMP que esteve realmente presente nesse processo.

Agradeço também a meu orientador Dr. Mauro Oliveira pela paciência e disposição em orientar-me, independente do horário ou local que dessem certo, ao meu co-orientador César Olavo e a meu amigo desde a infância, o Mestre Daniel Pordeus, que se dispôs em ser meu co-orientador externo.

Obrigado a todos vocês!

“Se alguém procura a saúde, pergunta-lhe primeiro se está disposto a evitar no futuro as causas da doença; em caso contrário, abstém-te de o ajudar”

Sócrates

RESUMO

A preocupação da população com a saúde, juntamente com o papel do governo brasileiro em oferecer meios para melhorar o ambiente de saúde, levou à origem do projeto LARIISA, um sistema inteligente responsável por criar soluções de tecnologia da informação para a área de saúde, objetivando auxiliar a administração pública de saúde à tomada de decisões, provendo melhores serviços à comunidade. Várias propostas no LARIISA são baseadas em sensibilidade ao contexto, caracterizando situações específicas para tomadas de decisão. Consegue-se a modelagem de contexto e representação do conhecimento ao explorar o paradigma das ontologias, possuidoras, também, de técnicas de inferência. Uma área interessante de ser explorada dentro da saúde é a geolocalização, pois, através dela, é possível identificar o local onde ocorrem agravos de doenças, por exemplo. Com a localização onde as doenças ocorreram, se faz possível a criação de relatórios e medidas emergenciais para administradores da saúde pública. A necessidade de desenvolver aplicativos de localização sensíveis ao contexto, capazes de coletar informações sobre agravos de doença e auxiliar a prevenção de doenças, levou ao desenvolvimento do projeto COISA (Conselheiro Inteligente de Saúde da Plataforma LARIISA). Trata-se de um sistema que agrega o domínio de monitoramento e alerta ao LARIISA, oferecendo mais transparência no controle de endemias, facilidade na criação e divulgação de relatórios para a governança de saúde e o auxílio à prevenção para o cidadão. O COISA possui dois subsistemas. O primeiro é uma aplicação web chamada GeoHS (*Geographic Health System*), responsável pelo cadastro de entidade no sistema, geração de relatórios e núcleo de inferência. O segundo sistema é o COISA Mobile (COBILE), voltado para dispositivos móveis e responsável por colher informações de usuários comuns e especialistas por meio de seus aparelhos.

Palavras-chave: saúde; monitoramento e alertas de doenças; *openEHR*; ontologia; LARIISA.

ABSTRACT

The concern of the population to health, along with the role of the Brazilian government to provide means to improve the health environment, led to the origin of LARIISA project, an intelligent system responsible for creating information technology solutions for healthcare, aiming assist the government in health decision-making, providing better services to the community. Several proposals in LARIISA are based on context awareness, featuring specific situations for decision-making. Achieved by the context modeling and representation of knowledge to explore the paradigm of ontologies, possessing also inference techniques. An interesting area to be explored within the health is the geolocation, because through it, you can identify where occur aggravations of diseases, for example. With the location where disease occurred, the reporting and emergency measures for administrators of public health is possible. The need to develop location-based applications context-sensitive, able to collect information about health hazards and help prevent disease led to the development THING project (Health Smart Advisor of LARIISA Platform). It is a system that combines monitoring domain and alerts the LARIISA, providing more transparency in the control of endemic diseases, ease the creation and dissemination of reports for health governance and aid for prevention for citizens. The THING has two subsystems. The first is a web application called GEOHS (Geographic Health System), the entity responsible for registration in the system, reporting and core inference. The second system is the THING Mobile (COBILE), focused on mobile devices and spoon responsible for ordinary users of information and experts through their devices.

Key words: health; monitoring and alerts of diseases; *openEHR*; ontology; LARIISA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACE	Agentes de Controle de Endemias
ACS	Agentes Comunitários de Saúde
AJAX	*Asynchronous JavaScript with XML*
API	Interface de Programação de Aplicações
BVSMS	Biblioteca Virtual em Saúde do Ministério de Saúde
CKM	*Clinical Knowledge Governance*
CDI	*Contexts and Dependency Injection*
CID-10	Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde
CLARIISA	*Framework* Sensível ao Contexto Baseado em Geolocalização para Sistema de Governança em Saúde
COBILE	COISA *Mobile*
COISA	Conselheiro Inteligente de Saúde do Projeto LARIISA
CONASS	Conselho Nacional de Secretários de Saúde
FII	Ficha Individual de Investigação
FIN	Ficha Individual de Notificação
GeoHS	*Geographic Health System*
GPS	Sistema de Posicionamento Global
GUI	Interface Gráfica do Usuário
HTML	*HyperText Markup Language*
JSF	*JavaServer Faces*
LARIISA	Laboratório de Redes Inteligentes e Integradas de Saúde
MVC	*Model View Controller*
OWL	*Web Ontology Language*
OWL DL	*Web Ontology Language Description Logics*
PSF	Programa Saúde da Família
RDF	*Resource Description Framework*
RES	Registro Eletrônico de Saúde
REST	*Representational State Transfer*

SIMDA	Sistema de Monitoramento Diário de Agravos
SINAN	Sistema de Informação de Agravos de Notificação
SOAP	*Simple Object Access Protocol*
SUS	Sistema Único de Saúde
UECE	Universidade Estadual do Ceará
UFC	Universidade Federal do Ceará
UML	*Unified Modeling Language*
URL	Localização-Padrão de Recursos (*Uniform Resource Locator)*
W3C	*World Wide Web Consortium*
XHTML	*eXtensible Hypertext Language*
XML	*Extensible Markup Language*
XSD	XML *Schema*

Sumário

1. INTRODUÇÃO

Uma informação de interesse social é o conhecimento de ocorrências de doenças em uma determinada região onde uma determinada pessoa se encontra. Esta informação pode auxiliar nas precauções necessárias para reduzir os riscos de se contrair alguma doença. A ocorrência de uma doença é chamada de *evento* (BRASIL, 2011). Porém, tal informação, quando existe, geralmente não se encontra acessível ao público em um formato de fácil compreensão.

Para que esse tipo de informação seja disponibilizado de uma maneira mais compreensível, seria necessário acesso a dados existentes no Sistema Único de Saúde (SUS) e a compilação desses dados.

Na cidade de Fortaleza, por exemplo, o Sistema de Monitoramento Diário de Agravos (SIMDA) (SIMDA, 2013), é mantido pela Prefeitura Municipal e alimentado por um sistema nacional, o Sistema de Informação de Agravos de Notificação (SINAN) (SINAN, 2014), mantido pelo Ministério da Saúde. O SINAN contém informações relevantes sobre doenças epidemiológicas, como dengue e leptospirose, bem como o local onde os agravos dessas doenças ocorrem.

Assim, para obter-se informação sobre as ocorrências de alguma doença, é preciso se ter acesso aos dois sistemas – SIMDA e SINAN – além de se realizar pesquisas sobre uma doença específica. No caso do SINAN, trata-se de um sistema desktop, não permitindo a portabilidade para dispositivos móveis, por exemplo.

Endemia é o conceito utilizado quando da ocorrência habitual de uma doença ou de um agente infeccioso em uma determinada área geográfica (BRASIL, 1977). O SINAN possui informações pertinentes que podem ser utilizadas para identificar endemias em um local, sendo uma fonte de informações para a gerência pública de saúde. Seus dados servem de subvenção para descoberta da localização de endemias.

Como estratégia para conter endemias, existe planejamento por parte do governo brasileiro, onde se define como cada um dos estados e Distrito Federal deve realizar o devido controle (BRASIL, 2001). A descentralização de ações de saúde da abrangência do Sistema Único de Saúde (SUS) (BRASIL, 2014a) também pertence ao plano proposto pelo Estado.

Além do controle de endemias por parte dos estados e Distrito Federal, também existem outras formas de descentralização de serviços do SUS. O Programa Saúde da Família (PSF) é um dos principais exemplos de descentralização desses serviços (BRASIL, 2014b).

Com o PSF, o atendimento é levado às residências dos pacientes, e não apenas em hospitais ou postos de saúde, o que torna o sistema de saúde mais abrangente, complexo e com um volume maior de informação devido a diversidade de locais de entrada de dados. Em virtude do volume maior de informações coletadas advindas do PSF, há mais precisão na detecção em ocorrências de endemias.

Para controle de endemias, como a malária, por exemplo, são necessárias ações conjuntas entre a população, os agentes de saúde e a governança de saúde. É nessa perspectiva que surgem os Agentes Comunitários de Saúde (ACS) e os Agentes de Controle de Endemias (ACE). Esses agentes conhecem os problemas e a rotina das comunidades, e, com isso, podem ajudar a amenizar as dificuldades encontradas em cada região no combate às doenças, não só da malária, mas endemias em geral (BRASIL, 2014c).

No âmbito do auxílio a agentes comunitários, ao público em geral e até para especialistas em estratégia e governança na área de saúde, se torna relevante a possibilidade de concepção de um sistema capaz de associar os dados sobre as doenças de cada região, contidos no SINAN, com dados relativos a cada paciente.

O histórico dos registros importantes de pacientes pode ser obtido utilizando o modelo de Registro Eletrônico de Saúde (RES) (BVSMS, 2015). Os dados contidos no RES armazenam os antecedentes do paciente, bem como as vacinas tomadas, doenças, eventos e agravos.

A integração entre o SINAN e o RES pode ser bem explorada em um sistema sensível ao contexto, capaz de identificar situações de acordo com parâmetros pré-definidos e, logo após, tomar alguma decisão. Um exemplo seria um cidadão que chega à noite em sua casa e as luzes acendem assim que ele passa pelos cômodos. O contexto então é parametrizado na presença do cidadão nos cômodos e no horário definido para diferenciar quando é noite. A decisão nesse caso é simples: acender ou não a luz.

Objetivando auxiliar o LARIISA (OLIVEIRA, 2010), no domínio de inteligência clínica-epidemiológica, sugeriu-se um sistema capaz de recuperar dados de doenças de um local e, baseado nos antecedentes de um usuário ou paciente, alertar usuários e entidades responsáveis pela governança de saúde sobre os possíveis riscos naquele local.

A partir da sugestão citada anteriormente, definiu-se o objetivo geral deste trabalho: especificar e implementar um sistema de localização para monitoramento e alertas de doenças, baseado no perfil epidemiológico de regiões geográficas e no histórico epidemiológico dos usuários. O sistema proposto foi nomeado Conselheiro Inteligente de Saúde do Projeto LARIISA (COISA).

A partir da localização atual do usuário ou de um destino do usuário, o COISA emite sugestões e alertas, explicando os riscos de ir para um determinado local. Para tanto ele utiliza o histórico de saúde do usuário por meio do Registro Eletrônico de Saúde (RES) (BVSMS, 2015) e as informações contidas nas bases de dados locais (SIMDA/SINAN, no caso de Fortaleza).

Os objetivos específicos do trabalho seguem abaixo:

- Inclusão do modelo de monitoramento e alertas do COISA na arquitetura do LARIISA;
- Especificação formal da solução proposta pelo COISA;
- Produção dos artefatos de software necessários para o desenvolvimento dos sistemas que compõem o COISA;
- Definição das tecnologias utilizadas para o desenvolvimento e a implantação dos sistemas;
- Implementação do protótipo em um ambiente de teste para consolidar o monitoramento e os alertas com conselhos no projeto LARIISA.

Como o projeto é voltado para a área de saúde, regras de negócio foram elaboradas e validadas por opinião médica, resultando na criação de um modelo das entidades integrantes do sistema, compondo diagramas de classes e de casos de uso. Posteriormente, deu-se início à implementação do sistema, utilizando as especificações e tecnologias descritas na documentação. Como resultado, foram gerados os artefatos da solução, como os documentos e os executáveis necessários

para a implantação do sistema em um ambiente de produção, além de uma análise do que pode ser continuado ou melhorado no projeto.

A dissertação é organizada da seguinte forma: o capítulo 02 apresenta uma descrição do estado da arte e referencial teórico, propondo uma visão do problema e o que se tem feito recentemente para resolvê-lo. O capítulo 03 trata da descrição do sistema proposto nesta dissertação, especificando os requisitos funcionais necessários para o desenvolvimento dos módulos. Também são apresentados diagramas de Linguagem de Modelagem Unificada (UML) para uma visão geral das entidades. Além disso, cenários são ilustrados para cada funcionalidade do sistema. Para detalhar a implementação do sistema, o capítulo 04 traz informações de requisitos não-funcionais, metodologia de desenvolvimento e partes relevantes na codificação do projeto COISA. Por último, no capítulo 05 é apresentada a conclusão, detalhando as contribuições relevantes deste trabalho e as análises do que se pode melhorar no projeto, além dos resultados obtidos.

2. FUNDAMENTAÇÃO TEÓRICA

Ao realizar o desenvolvimento do projeto COISA, foi imprescindível o conhecimento e a integração dos seguintes temas: sistemas sensíveis a contexto, ontologias, o projeto LARIISA e registros eletrônicos de saúde. Este capítulo apresenta os principais conceitos e a integração desses temas.

Em computação, há vários meios de se determinar uma situação. Uma solução viável para o projeto COISA foi relacionar sistemas sensíveis a contexto de localização com ontologias.

O projeto LARIISA é apresentado como a plataforma para o projeto COISA. Uma das funcionalidades do LARIISA é dar suporte à realização de inferência na criticidade de uma situação, fazendo uso de ontologias para decidir sobre a classificação da emergência de uma situação específica na área da saúde. Portanto, dentre as especificidades da ontologia, o projeto COISA focou em: classificação e inferência.

O COISA apresenta, também, uma forma para recuperar prontuários e históricos de pacientes e caracterizar uma situação na área de saúde de forma automatizada: Registro Eletrônicos de Saúde (RES). Para tanto, foi escolhida a arquitetura do *openEHR*[1], responsável por separar conceitos de domínios clínicos de saúde do desenvolvimento de software. Assim, o projeto COISA usa o *openEHR* para identificar cenários relacionados a saúde.

O capítulo de fundamentação teórica finaliza com o estudo do SINAN, sistema do governo brasileiro responsável por divulgação dos dados de agravos de doenças e notificações de eventos. Os dados do SINAN foram utilizados no projeto COISA juntamente com os RES para determinar riscos para a população em uma região.

2.1. Contexto

O conceito de contexto em computação, definido por Schilit, Adams e Want

[1] *openEHR* é uma comunidade virtual que trabalha na interoperabilidade e computabilidade em saúde electronica (e-health). Seu principal foco são os registros eletrônicos de saúde dos pacientes (OPENEHR, 2015a), além de ter uma arquitetura que separa bem conceitos clínicos de processos de desenvolvimento de software.

(1994) está relacionado ao local, à coleção de pessoas ao redor, hospedeiros e dispositivos acessíveis, além das mudanças desses fatores ocorridas durante o tempo. Segundo esses autores, há três aspectos relevantes a considerar: "Onde você está", "quem está com você", e "quais os recursos que estão nas proximidades?". Partindo dessas questões, o contexto é dividido em:

- **Contexto computacional**: processadores disponíveis, dispositivos acessíveis por entrada de dados e tela, impressoras, capacidade de rede, conectividade, custos de computação e comunicação, e largura de banda;
- **Contexto de usuário**: local do usuário, coleção de pessoas próximas, perfis de usuário e situação social;
- **Contexto físico**: luminosidade, temperatura, barulho, nível de umidade e condições de tráfego.

Em Liu (2011) são listadas várias definições de contexto. Uma dessas definições exibe contexto do ponto de vista da Computação Ubíqua. Com a Computação Ubíqua, o contexto é focado nas pessoas e torna o computador uma parte invisível nas nossas vidas. Com a máquina sendo invisível ao usuário, se consegue oferecer um serviço mais eficiente e com a máxima facilidade. Dessa perspectiva voltada para pessoas, se define contexto como:

- **Contexto de usuário**: algo relacionado ao usuário, incluindo informações dinâmicas (usuário atual, histórico de locais, emoção atual do usuário, relacionamentos ou contatos com colegas ou amigos, etc) e informações estáticas (situação social, informação pessoal do usuário, hábitos do usuário e preferências do usuário);
- **Contexto físico**: informação do ambiente físico (luminosidade, barulho, temperatura, nível de umidade e condições de tráfego) e informações sobre dispositivos físicos (bateria do dispositivo, memória, tamanho e tipo de tela, sistema operacional do terminal, entrada e saída de método, recursos próximos, etc);
- **Contexto de rede**: capacidade de rede, conectividade, custos de comunicação e comunicação, largura de banda, entre outros.

Outra definição é dada por Dey (1999): contexto é uma informação a ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação.

2.1.1. Sistemas sensíveis ao contexto

Sistemas Sensíveis ao Contexto se adaptam de acordo com as variáveis de ambiente envolvidas, como o local de uso, pessoas próximas, hospedeiros e dispositivos disponíveis, bem como as mudanças dessas variáveis no decorrer do tempo. Tais sistemas podem examinar o ambiente e reagir a mudanças para o ambiente (SCHILIT, ADAMS e WANT, 1994).

Sensibilidade ao contexto é, portanto, a habilidade de um programa ou um dispositivo de computador detectar, sentir, agir e responder a aspectos do ambiente, como a localização, luminosidade, temperatura ou identidade do usuário (SCHILIT; THEIMER, 1994).

Dentre as diversas variáveis possíveis para identificar o contexto de uma situação, o projeto COISA focou na variável "localização", a qual auxilia a representação das coordenadas geográficas do usuário a partir da triangulação de antenas de um celular, ou do Sistema de Posicionamento Global (GPS) de um dispositivo móvel.

2.1.2. Sensibilidade de contexto baseado na localização

Um dos contextos de maior importância para o projeto COISA é o de localização geográfica ou geolocalização. Neste tema existe o conceito de áreas demarcadas ou áreas de interesse (*geofences*).

Geofencing (delimitação geográfica) aplicada em computação combina sensibilidade da localização atual do usuário com a sensibilidade das características nas proximidades, definidas como proximidades do usuário para possíveis locais de interesse.

Para marcar um local de interesse, é preciso especificar a latitude e a longitude do local. Para ajustar a aproximação para o local, basta adicionar o raio. Os atributos latitude, longitude e raio definem uma *geofence* (área de interesse).

No desenvolvimento de aplicativos para o sistema operacional *Android* (ANDROID, 2014), é possível verificar múltiplas áreas de interesse, simultaneamente (GEOFENCE, 2014).

A figura 01, a seguir, ilustra uma área de interesse, representada em um mapa com visualização 3D, utilizando a Interface de Programação de Aplicações (API) do *Google*.

Figura 01 - Área demarcada, representada no mapa (GRAPE STUDIOS, 2014)

Na figura 01, o ponto circular vermelho representa a latitude e a longitude do ponto de interesse, enquanto a área verde é o círculo definido a partir das coordenadas e do raio de uma *geofence*.

Para cada área demarcada é possível solicitar os *Serviços de Localização*[2] para enviar eventos de entrada, saída, ou entrada e saída simultaneamente daquela área.

[2] *Serviços de Localização* são serviços providos pela API do Google para o sistema operacional *Android* e trata uma *geofence* como uma área, ao invés de pontos e proximidade. Isso permite detectar quando algum usuário entra ou sai de uma determinada área de interesse.

Também há alternativa de limitar o tempo que uma *área de interesse* é válida, especificando um tempo de expiração (validade) em milissegundos. No momento em que o tempo expirar, os *Serviços de Localização* removem a *área de interesse* automaticamente.

Os principais tipos de modelagem de contexto são apresentados no próximo tópico.

2.1.3. Modelagem de Contexto

Importantes abordagens de modelagem de contexto em computação ubíqua são explanadas por Strang e Linnhoff-Popien (2004): modelo chave-valor, modelos de esquemas de marcação, modelos gráficos, modelos orientados a objetos, modelos baseados em lógica e modelos baseados em ontologias.

Dentre as abordagens citadas, este trabalho tomou como base o modelo de ontologias devido à sua capacidade de compartilhamento de conhecimento, capacidade de reuso do conhecimento e, principalmente, de seu potencial de inferência.

2.2. Ontologias

2.2.1. Definições

Segundo Guarino e Giaretta (1995), existem 7 interpretações para a palavra "ontologia":

1. Ontologia como uma disciplina filosófica;
2. Ontologia como um sistema conceitual informal;
3. Ontologia como uma consideração semântica formal;
4. Ontologia como uma especificação de uma conceitualização;
5. Ontologia como uma representação de um sistema conceitual via teoria lógica:
   5.1. caracterizada por propriedades formais específicas;
   5.2. caracterizada somente por suas propostas específicas;

6. Ontologia como o vocabulário usado por uma teoria lógica;
7. Ontologia como uma meta-especificação de uma teoria lógica.

Dentre as interpretações listadas, uma distinção importante está entre a primeira interpretação e todas as outras.

Ontologia é uma descrição explícita formal de conceitos em um domínio de discurso (**classes** ou conceitos), propriedades de cada conceito descrevendo características e atributos dos conceitos (*slots*, *roles* ou **propriedades**), e restrições (***facets*** ou *role restrictions*). A ontologia juntamente com as instâncias das classes que a compõem formam a **base de conhecimento** (NOY e MCGUINNESS, 2014).

De acordo com Uschold e Gruninger (1996), "Ontologia" é um termo usado para se referir ao entendimento compartilhado e algum domínio de interesse. Uma ontologia implica ou engloba algum tipo de visão de mundo por perspectiva de um determinado domínio. Esta visão de mundo é concebida como um conjunto de conceitos (entidades, atributos e processos, por exemplo), suas definições e seus inter-relacionamentos. Tal visão é também conhecida como conceitualização.

O grau de formalismo de como um vocabulário é criado e como seu significado é especificado varia consideravelmente. Uschold e Gruninger (1996) definem os seguintes níveis de formalismo do vocabulário:

- **altamente informal**: expressa na linguagem natural;
- **semi-informal**: expressa na linguagem natural em uma forma restrita e estruturada;
- **semi-formal**: expressa em uma linguagem artificial definida formalmente;
- **rigorosamente formal**: definida em termos com semânticas formais, teoremas e provas de tais propriedades como solidez e complitude.

2.2.2. Classificações

Uschold e Gruninger (1996) ainda definem os papéis da ontologia, classificando-os em:

- **Comunicação**: para facilitar a comunicação entre pessoas e organizações;
- **Interoperabilidade**: entre sistemas. Exemplo: usar ontologia como uma interlíngua para unificar diferentes linguagens e ferramentas de software.
- **Engenharia de software**: ontologia pode facilitar o processo de construção e manutenção de software nos seguintes pontos:
    - **Reusabilidade**: ontologia quando representada em uma linguagem formal pode ser um componente reutilizável em um sistema de software;
    - **Confiabilidade**: uma representação formal facilita avaliação de consistência automática;
    - **Especificação**: ontologia auxilia o processo de identificar requisitos para construir a especificação para um sistema de TIC.

Guarino (1998) classificou a ontologia em quatro tipos, baseado no nível de generalidade:

- ***top-level***: descreve conceitos gerais, como espaço, tempo, dificuldade, objeto, evento e ação, independentes de um problema ou domínio particular;
- ***domain***: vocabulário relacionado ao domínio genérico (como medicina ou automóveis);
- ***task***: tarefa genérica ou alguma atividade (como diagnóstico de saúde ou venda), especializando termos definidos na ontologia *top-level*;
- ***application***: descreve conceitos que dependem tanto de ontologias de *domain* quanto de *task*, sendo especializações de ambas ontologias relacionadas.

2.2.3. Motivação do uso de ontologias

De acordo com Noy e McGuinness (2014), os principais motivos para se desenvolver ontologia são:

- Compartilhar entendimento comum da estrutura de informação entre pessoas e agentes de software;
- Disponibilizar reuso de conhecimento de domínio;
- Criar hipóteses explícitas de domínio;
- Separar conhecimento de domínio de conhecimento operacional;
- Analisar conhecimento de domínio.

A possibilidade de usar uma máquina de inferência para derivar verdades adicionais sobre conceitos já modelados é o principal motivo para Guermah et. al. (2013) implementar aplicações baseadas em ontologias.

Ko e Sim (2008) justificam o uso de modelagem em ontologias porque precisam desenvolver aplicações que utilizam computação sensível ao contexto. O modelo do contexto pode ser estruturado pelo uso da ontologia e pode gerenciar através do contexto de acesso semântico.

2.2.4. Linguagens para ontologias

A *Web Ontology Language* (OWL) foi criada para o desenvolvimento de ontologias na web, a partir das seguintes linguagens citadas em McGuinness & Harmelen (2004):

- **XML (eXtensible Markup Language)**: fornece uma linguagem para documentos estruturados, sem impor restrições semânticas sobre o significado desses documentos;
- **XML Schema**: linguagem para restringir documentos XML, validando seus campos, como também estender XML com *datatypes* (tipos de dados);
- **RDF (Resource Description Framework)**: modelo de dados para objetos (chamados de recursos) e relações entre eles, fornecendo uma semântica simples para esse modelo de dados, que pode ser representado em XML;

- **RDF Schema**: vocabulário para descrever propriedades e classes de recursos RDF, com uma semântica para generalização de hierarquias das propriedades e classes;
- **OWL**: adiciona mais vocabulário para descrever propriedades e classes: relação entre classes (disjunção, por exemplo), cardinalidade, igualdade, tipagem rica de propriedades, características das propriedades e classes enumeradas.

A figura 02 mostra a arquitetura da web semântica proposta por Tim Berners Lee (BERNERS-LEE, 2005) e as várias linguagens utilizadas.

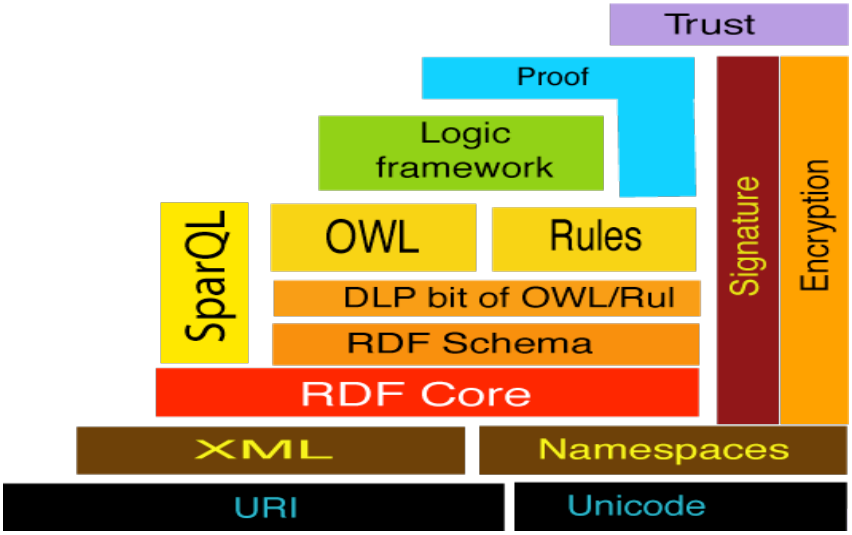

Figura 02 - Pilha de linguagens da Web Semântica (BERNERS-LEE, 2005)

OWL possui 3 sub-linguagens projetadas para comunidades específicas e usuários e implementadores. São elas:

- **OWL Lite**: dá suporte aos usuários que precisam de uma hierarquia de classificação e restrições simples. Oferece facilidade para migrar para tesauros e outras taxonomias. Possui baixa complexidade formal.
- **OWL DL**: dá suporte a usuários que necessitam da máxima expressividade possível, mantendo a completude computacional e decidibilidade. É baseada em lógica descritiva.
- **OWL Full**: para usuários que além da máxima expressividade necessitam de liberdade sintática sem garantias computacionais. Uma classe pode ser tratada simultaneamente como uma coleção de

indivíduos ou como um indivíduo próprio. Permite à ontologia aumentar o significado do vocabulário pré-definido (RDF ou OWL).

2.3. LARIISA

De acordo com (OLIVEIRA et al., 2010), o LARIISA é um *framework* para tomada de decisão da governança de sistemas públicos de saúde.

O *framework* foi projetado para atuar em 05 domínios de governança: gerência de conhecimento, normatização sistêmica, clínico-epidemiológica, administrativa e gerenciamento compartilhado.

Para a representação do conhecimento no LARIISA foram modeladas duas ontologias, descritas a seguir.

2.3.1. Modelo de contexto de saúde

A primeira ontologia do LARIISA é definida como informações de contexto local de saúde, onde é descrita a situação de uma entidade que interage com o sistema de governança. Essa entidade pode ser um agente de saúde, um especialista em saúde (médico) ou um paciente. Essas informações locais definem regras e podem formar o contexto global de saúde. O modelo de contexto local é definido na figura 03.

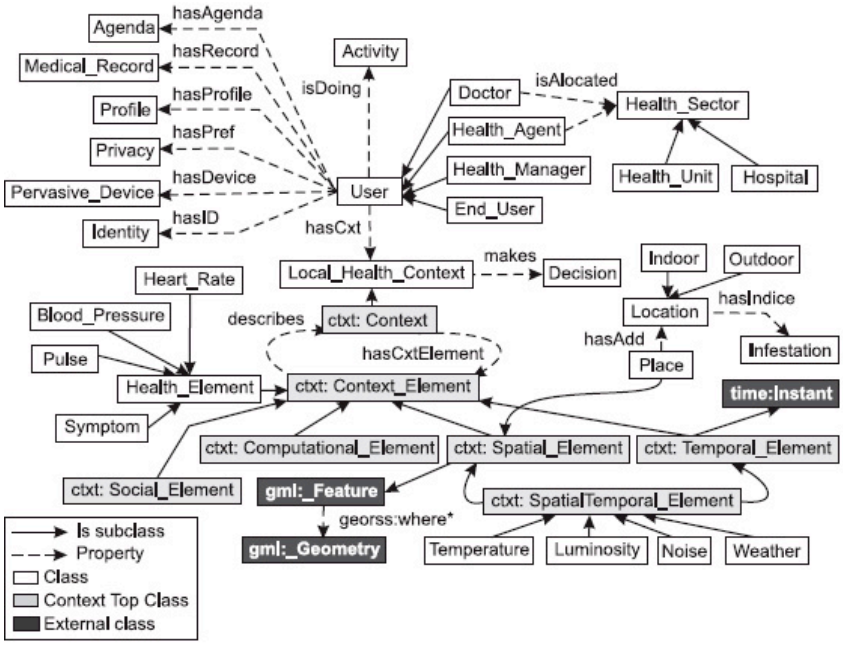

Figura 03 - Área demarcada, representada no mapa (GRAPE STUDIOS, 2014)

A segunda ontologia é a de contexto global de saúde, fazendo referência às informações de alto nível e sendo utilizada para tomada de decisão em governança de saúde. A figura 04 ilustra o modelo de contexto global.

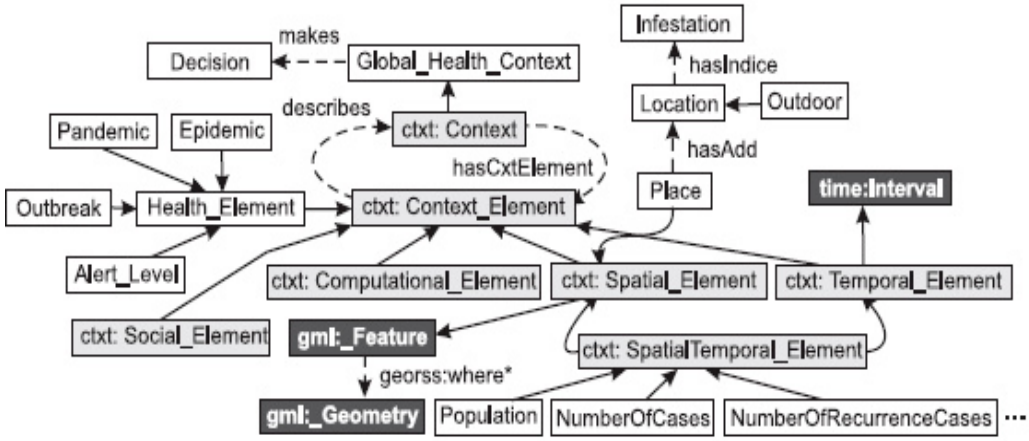

Figura 04 - Modelo Global de Saúde do LARIISA (OLIVEIRA et al., 2010)

Existem os níveis de alerta (*Alert_Level*), baseados nos elementos de saúde (*Health_Element*), caracterizados por pandemia, epidemia e surto. Este nível de alerta contextualiza gestores na governança sobre a gravidade de uma situação definida por elementos de contexto.

Os contextos de saúde local e global se classificam em:

- **espacial**: informações que caracterizam a situação da dimensão espacial (Por exemplo: localização, lugar, coordenadas GPS);
- **temporal**: informações que caracterizam a situação da dimensão do tempo (Por exemplo: instante, intervalo, período do dia, período do mês, período do ano, estação);
- **espaço-temporal**: informações que caracterizam a situação que depende da dimensão espacial e temporal (Por exemplo: clima, temperatura, ruído, luminosidade);
- **Social**: informações que caracterizam a situação dos relacionamentos sociais;
- **Computacional**: informações que caracterizam a situação de características computacionais (Por exemplo: configuração do dispositivos do usuário, como memória física disponível, velocidade de processamento, entre outros);
- **Elemento de saúde**: classifica o contexto da informação a partir do ponto de vista da saúde (Por exemplo: batimento cardíaco, pulso, pressão sanguínea, taxa de glicose, entre outros).

2.3.2. Domínios de governança

Os domínios de governança citados no início do capítulo são definidos como:

- **gerência de conhecimento**: domínio responsável por identificar, criar e representar experiências em cuidados de saúde;
- **normatização sistêmica**: participação de agentes públicos e gestores de saúde para a elaboração de leis relacionadas a saúde;
- **clínico-epidemiológica**: está relacionado ao conhecimento de doenças na área de saúde, com a saúde determinada através de fatores biológicos, sociais, econômicos, genéticos e de estilo de vida;
- **governança administrativa**: domínio responsável por gerenciar e alocar recursos (especialistas) para o trabalho em um determinado projeto;
- **gerenciamento compartilhado**: relacionado à forma de compartilhamento de conhecimento nos sistemas de saúde. Exemplo de conhecimento: competências do governo e dos sistemas estruturantes do governo.

2.3.3. LARIISA Core

A máquina geradora e processadora de contexto do LARIISA está no seu *core*. Chamado de **LARIISA Framework Core**, é dividido nos seguintes componentes:

- **Context Provider (CP)**: Provedor de contexto, responsável por coletar os dados brutos com contexto. Por exemplo: contexto de sensores móveis por meio de dispositivos móveis de agentes de saúde; Esses dados são enviados para o CA.
- **Context Aggregator (CA)**: o agregador de contexto é responsável por receber informações de contexto vindas dos provedores de contexto (CPs). Também tem a função de agregar contexto de alto nível representado por ontologia de contexto local de saúde.
- **Context Reasoner (CR)**: o inferidor (raciocinador) de contexto tem como função realizar inferências a partir de contexto local de saúde e gerar contexto global de saúde. Por exemplo, o CR pode inferir que determinada região tem uma endemia (contexto global de saúde) a partir de perfis epidemiológicos de uma região (contexto local de saúde).
- **QoC Evaluator (QoCE)**: o avaliador de qualidade de contexto determina a qualidade para cada conceito de contexto, auxiliando decisões de governança em saúde.
- **Service Adapter (SA)**: Responsável por realizar adaptação sensível ao contexto e executar regras de decisão local e global de saúde. Além disso, identifica a informação relevante para o contexto de saúde e classifica em um dos ciclos:
    - Processo de criação de conhecimento;
    - Processo de tomada de decisão em governança de saúde;
    - Ações e cuidados de saúde.
- **Context-Aware Service (CAS)**: recebe informações já filtradas pelo SA e executa suas funcionalidades baseadas nessas informações.

- **CAS Container (CasC)**: Um CasC representa um grupo de CAS. Um sistema que utiliza tomada de decisão contém um ou mais *containers* de serviços sensíveis ao contexto.
- **Query Adapter (QA)**: responsável pela manipulação de consultas contextuais, extraindo informações relevantes do repositório de contexto global de saúde.

A figura 05 mostra a arquitetura de integração dos componentes.

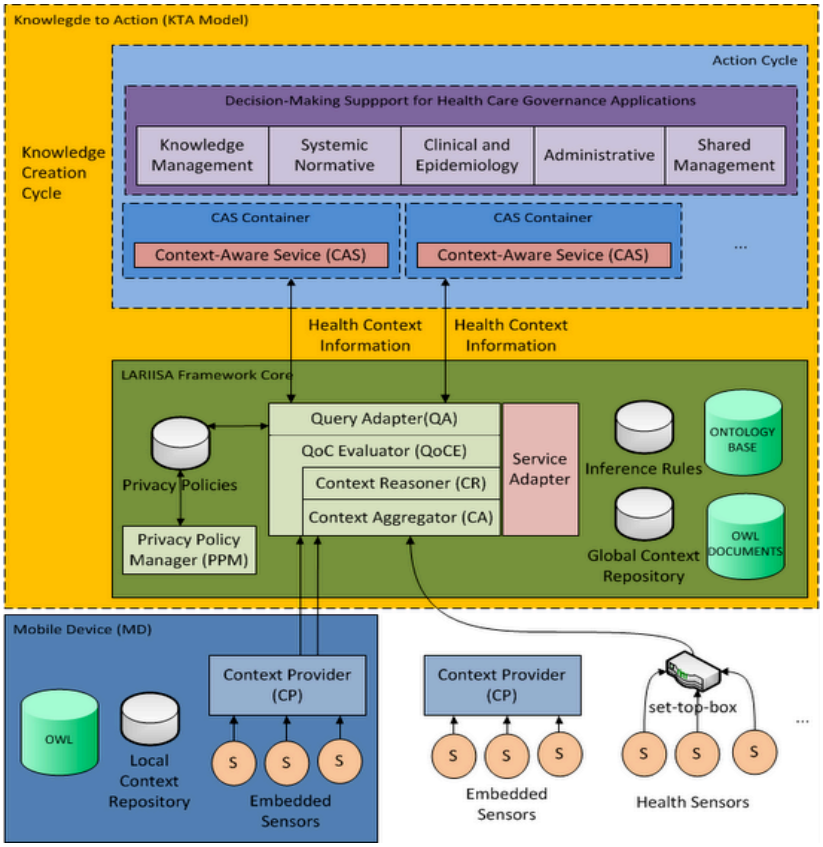

Figura 05 - Lariisa *Framework Core*

2.4. Registro Eletrônico de Saúde

O projeto COISA armazena os dados dos pacientes em uma estrutura de dados chamada Registro Eletrônico de Saúde (RES), um repositório de informações sobre o estado de saúde de um paciente, em uma forma computável

eletronicamente (ISO/TR 20514, 2005, p. 2). Para o desenvolvimento do RES, foi utilizada a arquitetura *openEHR*.

2.4.1. *OpenEHR*

O *openEHR* é uma comunidade que tem como objetivo padronizar os registros eletrônicos de saúde. Sua forma de tratar a informação clínica é independente do modelo computacional. Na prática, há uma separação lógica entre os profissionais da área de saúde e especialistas da computação (OPENEHR, 2015a).

O domínio médico é representado por modelos clínicos chamados arquétipos. Toda regra relacionada a doença, diagnóstico, precauções, prevenções, entre outras, é definida através dos arquétipos (OPENEHR, 2015a).

A figura 06 a seguir mostra a arquitetura do *openEHR*.

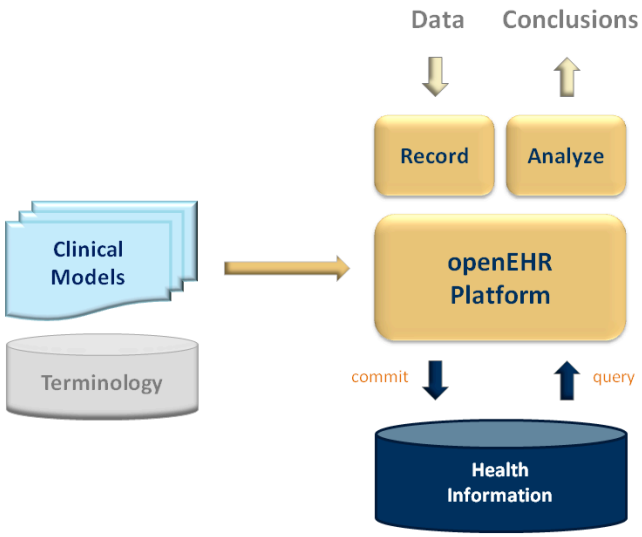

Figura 06 - Arquitetura do OpenEHR. (OPENEHR, 2015a)

O *Clinical Knowledge Governance* (CKM) trata a governança de conhecimento em saúde (CKM, 2015). Ele permite a modelagem e o compartilhamento de arquétipos, modelos e terminologias integrados ao *openEHR*.

Devido a independência entre a área de negócio e a área de Tecnologia da Informação, a criação de modelos clínicos se torna confortável para ambas as

partes. Para aumentar a produtividade do desenvolvimento de software, a arquitetura do *openEHR* fornece os *templates*, partes de código geradas automaticamente a partir de configurações baseadas nos arquétipos.

Sendo assim, aqueles que têm o *know how* na área de saúde (profissionais como: médicos, enfermeiros, etc.) passam a ser participantes efetivos e diretos na construção dos sistemas, fazendo com que as informações de um sistema *openEHR* possuam grande confiabilidade. O modelo que representa a maneira de interação, bem como a independência entre os papéis dos profissionais de saúde e dos desenvolvedores de software é ilustrado na figura 07 a seguir.

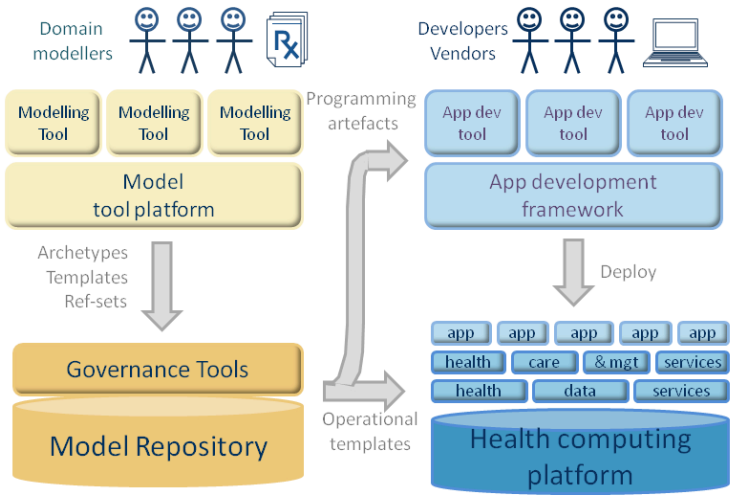

Figura 07 - Arquitetura de independência entre profissionais de saúde e desenvolvedores de software (OPENEHR, 2015a)

2.4.2. *openEHR* e ontologias

A arquitetura do *openEHR*, conforme explanado alhures, divide claramente o conhecimento do negócio de saúde do conhecimento de informática, fazendo com que cada um realize seu papel de maneira autônoma e independente. A ontologia, por sua vez, representa o conhecimento acerca de determinado domínio. No caso do *openEHR*, o domínio raiz é a *saúde*. Dessa forma, a Fundação *openEHR* trabalha no sentido de elaborar um modelo de informações de saúde que seja o

mais adequado para inferência de dados, apoio à decisão, tratamento médico, entre outros (OPENEHR, 2015b).

Uma análise demonstrando semelhanças entre ontologias e a arquitetura do *openEHR* é detalhada em (SANTOS, BAX e PESSANHA, 2010). De acordo com essa análise, a ontologia se divide em *ontologia central*, que descreve os conceitos de domínios e as relações entre eles; e os *compromissos ontológicos* (restrições), apresentados como conjunto de regras de domínio.

Da mesma forma, o modelo de referência em *openEHR*, padronizado pela norma ISO 13606 (ISO 13606, 2008), é um conjunto de classes (no paradigma *Orientação a Objetos*), as quais representam o domínio de RES. Por sua vez, os arquétipos, utilizados para criar modelos clínicos, são conjuntos de restrições que interligam o modelo de referência e os sistemas *openEHR*.

O modelo de informação da norma ISO 13606, em conjunto com os arquétipos, serve como base para uma aplicação fundamentada no conhecimento, quando comparado a outros modelos ontológicos. O modelo referencial da norma compreende classes desenvolvidas por especialistas em informática de vários países. Essas classes poderiam, por exemplo, ser representadas por meio de uma estrutura hierárquica dos Registros Eletrônicos de Saúde, numa ontologia desenvolvida com a ferramenta Protégé (PROTÉGÉ, 2014). A figura 08 representa tais semelhanças.

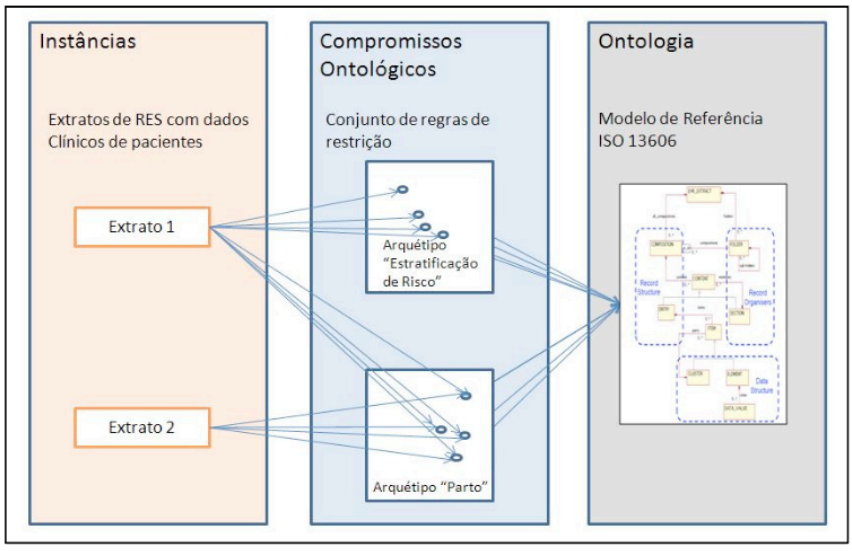

Figura 08 - Relação entre instâncias, arquétipos e modelo de referência (SANTOS, BAX e PESSANHA, 2010)

2.5. SINAN

O projeto COISA, além de armazenar os registros eletrônicos de saúde, precisa ser alimentado com informações sobre a localização de agravos ocorridos em uma determinada região. Com isso, passa a ser capaz de realizar inferências sobre quais áreas podem ser consideradas um risco para usuários do sistema. É por esse motivo, inclusive, que o COISA deve se utilizar de dados do SINAN.

O SINAN é responsável por receber dados referentes aos agravos de notificação, e ainda acerca de investigação de casos de doenças contidas na lista nacional de notificação compulsória (BRASIL, 2014d) (SINAN, 2014).

A classificação de eventos, agravos e doenças utilizada no SINAN, bem como quais as doenças e instituições responsáveis pelas notificações, estão disponíveis em (BRASIL, 2011).

O uso do SINAN possibilita explicações dos agravos de notificação compulsória e serve como indicação dos riscos aos quais a população está sujeita. Junto à informação dos agravos, é mantida a localização dos eventos, permitindo a identificação da situação epidemiológica em determinada área geográfica.

O SINAN precisa ser descentralizado, segundo normas do SUS. Portanto, a informação pode ser inserida diretamente no município, na região, ou no estado onde aconteceu o agravo. A maior parte das notificações é digitada pelas secretarias municipais de saúde. Se o município não for informatizado para efetivar a digitalização, os dados devem ser incluídos nas regionais de saúde (SINAN, 2014).

Quando há suspeita da ocorrência de problema de notificação compulsória, seja de interesse nacional, estadual ou municipal, é dever das unidades assistenciais preencherem a Ficha Individual de Notificação (FIN) para cada paciente. As secretarias de saúde utilizam a FIN para notificar a ocorrência de agravos. Além disso, existe a possibilidade de não haver suspeitas de eventos. Nesse caso é necessário o preenchimento do formulário de notificação negativa.

Após a emissão de uma FIN ou de uma ficha de negação de suspeita, é necessária a confecção de um documento afirmando ou negando a suspeita notificada. Tal documento é denominado Ficha Individual de Investigação (FII).

2.5.1. Situação do SINAN

O Conselho Nacional de Secretários de Saúde (CONASS) elaborou documento através do qual efetivou a avaliação do SINAN, enfatizando a complexidade e os problemas recorrentes do sistema (CONASS, 2013).

O SINAN necessita agregar informações de várias doenças e agravos de notificação obrigatória, além de adequar-se a diferentes padrões de alimentação de dados, e ser personalizado para aceitar solicitações das três esferas de gestão (CONASS, 2013).

Com a pretensão de amenizar os problemas gerados pela sua complexidade, o SINAN lançou vários *patches* (correções e melhorias em uma versão dos sistemas) durante o desenvolvimento. Porém, mesmo após as melhorias, novos erros eram gerados em decorrência das correções levadas a efeito sem testes de regressão (testes de funcionalidades que funcionavam antes de ajustes) (CONASS, 2013).

Vários programas compõem o SINAN: SINAN NET, SINAN Dengue, SINAN Influenza, SINAN Proadi e SINAN Relatórios. A figura 09 a seguir mostra o SINAN completo (ou SINAN Complexo, devido a seus vários sistemas).

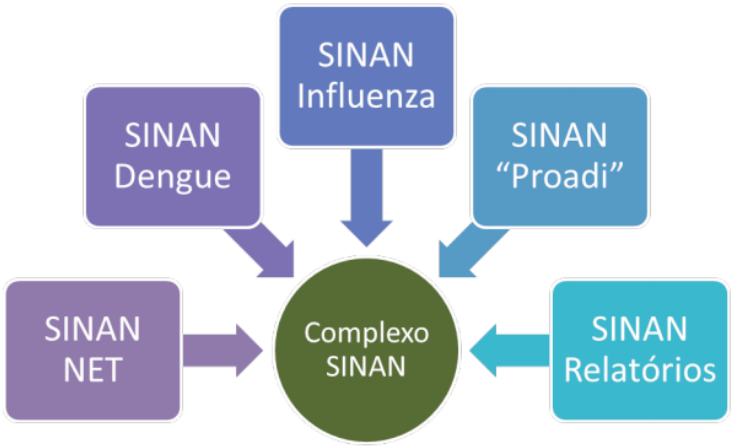

Figura 09 - SINAN Complexo (CONASS, 2013)

A seguir uma lista de limitações dos sistemas consideradas pelo CONASS:

- **SINAN NET**
  - Programação legada e mal estruturada;

  - Não possui relatórios epidemiológicos;
  - Lentidão ao ser conectado por vários usuários que simultaneamente tentam realizar exportação de dados, por exemplo.
- **SINAN Dengue**
  - Não possui versão desktop, nem móvel, para municípios desprovidos de serviços de internet;
  - A gestão da base de dados pelas Secretarias Estaduais e Municipais de Saúde não possui mecanismos de gestão local.
- **SINAN Influenza**
  - Não conta com ferramentas de exportação das bases de dados, além de apresentar as mesmas limitações do SINAN Dengue;
- **SINAN Proadi**
  - O sistema ainda está em desenvolvimento e no momento só pode ser implantado por especialistas na área de infraestrutura de computadores.

O SINAN Proadi está em fase de avaliação e testes. Ou seja, ainda não está disponível ao público em geral. Mesmo nesta nova versão do SINAN, ainda é preciso avaliar geração de relatórios, exportação de dados, níveis e perfis de acesso para os usuários, soluções desktop e móvel, e integração com outros sistemas de saúde (CONASS, 2015).

2.6. Trabalhos Relacionados

Este tópico apresenta as principais soluções encontradas na literatura para o desenvolvimento de sistemas, envolvendo sensibilidade ao contexto, localização, ontologias, desenvolvimento para web e para dispositivos móveis, voltados à área de saúde ou emergência pública.

Ao final do tópico, foi ilustrada uma comparação entre as soluções pesquisadas e o sistema COISA.

2.6.1. Sisa

O Sistema de Saúde Adaptado ao contexto de Gestão de Saúde (Sisa) utiliza a plataforma LARIISA, dá suporte à tomada de decisão, e foca na melhoria da qualidade de serviços prestados por agentes de saúde no combate a crises epidemiológicas, especialmente a dengue (SISA, 2011).

Os principais interessados ao Sisa são: Gestores de saúde, Agentes de Saúde Comunitária, Cidadãos e Administradores do sistema. Para atender às necessidades desses atores, o sistema é dividido em três módulos:

- **Módulo TV**: Captura informações das famílias. Esse aplicativo provê a entrada de dados para o módulo Web;
- **Módulo Mobile**: Aplicativo de uso dos Agentes de Saúde. O programa fornece agenda das visitas a serem realizadas nas residências, recebe notificações de urgência e gera contexto;
- **Módulo Web**: Sistema Web que possibilita ao gestor de saúde avaliar a situação epidemiológica de dengue, além de poder consultar mapas epidemiológicos e estatísticas que auxiliam na tomada de decisão.

A figura 10 a seguir contém a interação entre os três módulos.

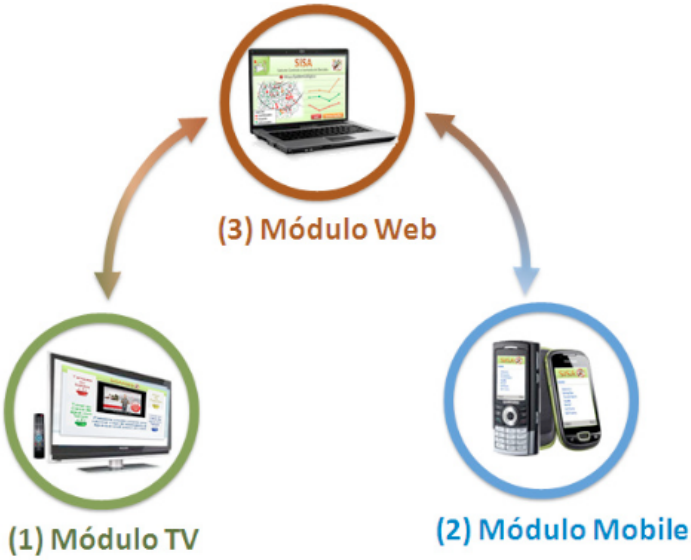

Figura 10 - Interação entre os três módulos do Sisa (SISA, 2011)

O fluxo de atividades do Sisa começa no Módulo TV, onde o cidadão comum é responsável por interagir com a TV, criando a base de conhecimento do Sisa. As regras contidas no Módulo Web são executadas sobre os dados de entrada, verificando, assim, se existem crises epidemiológicas e possivelmente gerar evento de alerta para o agente de saúde. O agente pode confirmar ou descartar a presença do foco (SISA, 2011). A seguir, na figura 11, está o diagrama do fluxo de atividades do Sisa.

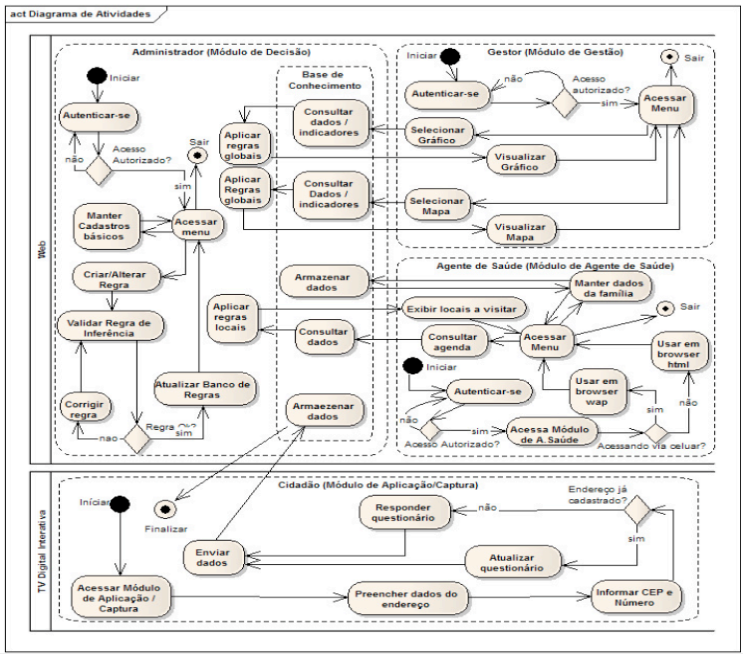

Figura 11 - Diagrama de fluxo de atividades do Sisa (SISA, 2011)

O motor de regras do Sisa é baseado no Drools (DROOLS, 2015). Uma das vantagens do Drools é a independência do banco de dados. As regras são armazenadas em arquivos específicos da linguagem (extensão *drl* - Drools Rule Language).

2.6.2. CLARIISA

O *Framework* Sensível ao Contexto Baseado em Geolocalização para Sistema de Governança em Saúde (CLARIISA) usa sensibilidade ao contexto e é baseado em geolocalização para um sistema de governança em cuidados de saúde

(GARDINI et. al., 2013). Portanto, fornece uma base relevante para o projeto COISA: sensibilidade ao contexto e geolocalização com dispositivos móveis.

A arquitetura do CLARIISA é uma extensão do modelo do LARIISA. É segmentada em 03 módulos:

- **Aquisição de dados**: através de informações prestadas pelos usuários do sistema, e de sensores de dispositivos móveis;
- **Processamento dos dados**: Associa e organiza os dados coletados para fornecer uma estrutura preparada para publicação no banco de dados do LARIISA;
- **Publicação**: responsável por salvar o conteúdo na base de dados do LARIISA. Aplicações podem utilizar a publicação para filtrar dados que servirão aos gerentes da área de saúde.

A figura 12 representa os 03 componentes citados:

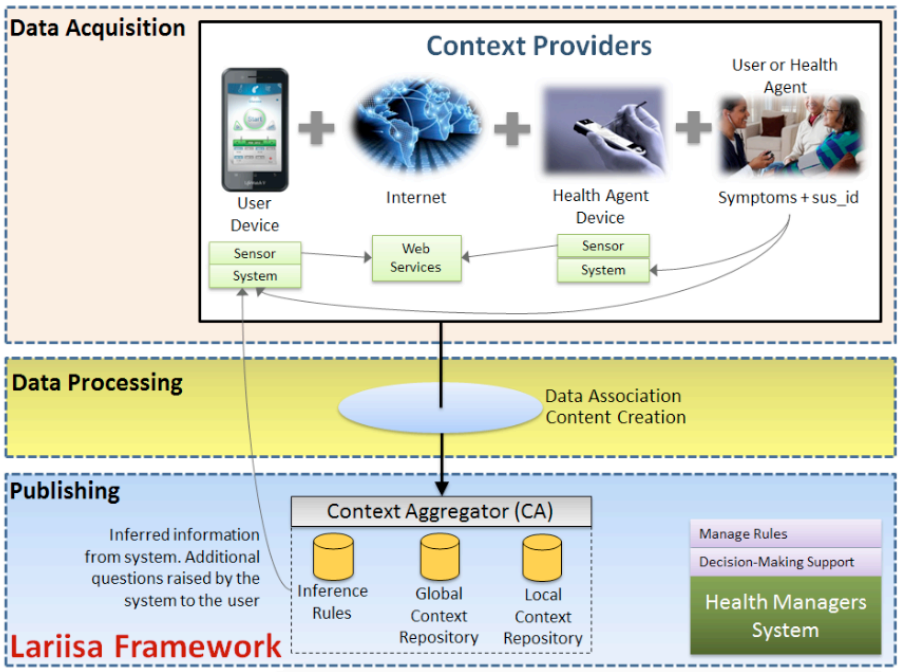

Figura 12 - Arquitetura do CLARIISA (GARDINI, 2013)

A aquisição de dados no protótipo do CLARIISA está sendo desenvolvida para o sistema operacional Android, assim como o protótipo para dispositivos móveis detalhado nesta dissertação.

O sistema de autenticação do CLARIISA utiliza o SUS ID (Identificador de Usuário do Sistema Único de Saúde) para identificação do usuário. É o mesmo atributo utilizado pelo governo no Cadastro do SUS.

2.6.3. INSIGMA

O *Intelligent System for Global Monitoring Detection and Identification of Threats* (INSIGMA) é um projeto que tem como objetivo desenvolver um sistema de monitoramento complexo que permite identificar objetos no ambiente monitorado e, baseado em informação armazenada e algoritmos avançados, identifica ameaças relacionadas ao tráfego e ao comportamento suspeito de pessoas. Por usar ontologias em seu desenvolvimento, esse sistema é referência para a modelagem de ontologias criadas no sistema COISA.

O sistema desenvolvido no INSIGMA é bem empregado para gerenciamento de tráfego, planejamento de rotas para usuários individuais e serviços de segurança pública (GLEBA et al., 2012).

A arquitetura do INSIGMA usa ontologia para descrever elementos e modelar entidades. As ontologias bem estruturadas facilitam a entrega de conhecimento na base do processo de inferência, que se baseia no fluxo do subsistema de monitoramento de tráfego. No INSIGMA, a ontologia é útil para identificar automaticamente ameaças e perigos de trânsito, além de automatizar a criação de notificações para serviços de segurança pública, como polícia, bombeiros, etc.

Um dos módulos desenvolvidos no INSIGMA é o *Event Notification Service* (ENS), responsável por criar, transmitir e processar mensagens curtas contendo informações sobre os eventos criados pelos usuários. No ENS existem o módulo de reportagem (localizado no aplicativo cliente) que permite a comunicação de acidentes e outros eventos para usuários longe do local da área de monitoramento, e o módulo de notificação (localizado no lado servidor).

Com a intenção de automatizar a classificação de eventos e ameaças, o ENS aplica o modelo de ontologia A-Box e verifica a descrição do evento entregue pelo módulo de reportagem. O módulo servidor se torna apto a classificar o evento e gera a notificação necessária para os serviços de segurança pública. O ENS também pode ser útil para exportar dados geográficos que podem ser representados em um

mapa com as dificuldades ou obstáculos da estrada, além de deixar possível a criação de rotas baseadas nos perigos e ameaças do trânsito.

A estrutura das mensagens utilizadas no ENS é composta basicamente pelo tipo, hora, e geolocalização do evento, bem como o identificador do usuário ou do equipamento. Cada mensagem é codificada em XML.

Uma das partes mais relevantes no INSIGMA, para esta dissertação, é a arquitetura do *Reasoning Module* (RM), que faz parte do ENS e foi desenvolvido utilizando programação Java e Webservices com implementação do *Simple Object Access Protocol* (SOAP). O conhecimento inferido tem como base os modelos de ontologia T-Box e A-Box, representando as taxonomias das classes que descrevem os conceitos e as instâncias, respectivamente.

O desenvolvimento de ontologias utilizadas pelo RM foi feito com o *Protégé* 3.6.4 (PROTÉGÉ, 2014), que dá suporte ao *Protégé* OWL API, este último responsável pela geração de código Java, crucial para ontologias *run-time* (em tempo de execução). A ontologia principal é a *InsigmaEventOntology*, responsável por chamar o método *sendEventDescription* usado para fazer a requisição ao RM. A figura 13 mostra a taxonomia da ontologia.

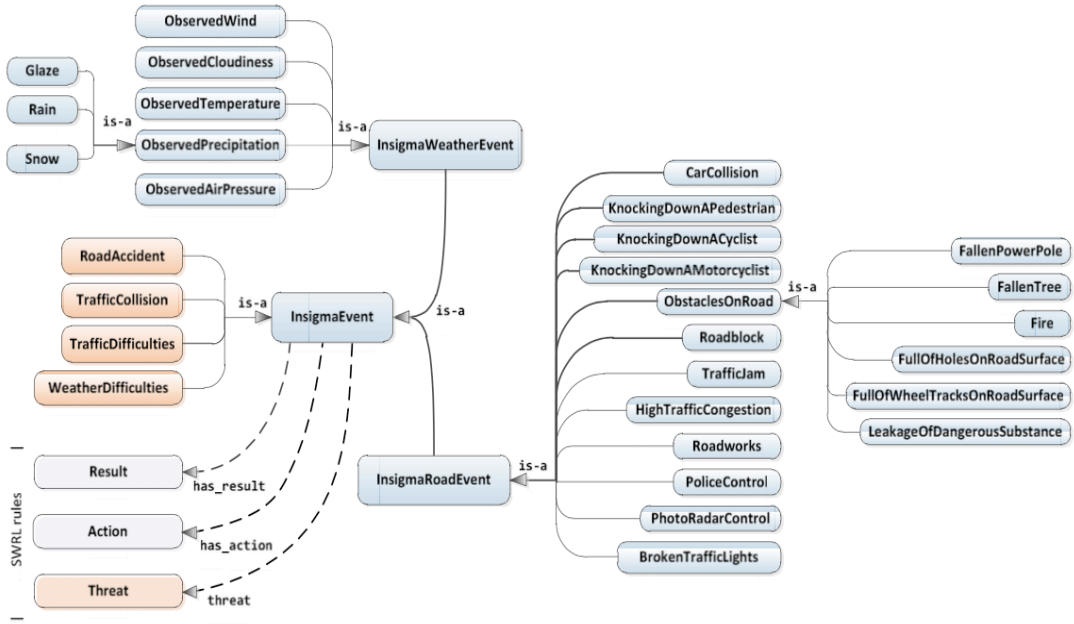

Figura 13 - Taxonomia da Ontologia de Eventos do INSIGMA (GLEBA et al., 2012)

A arquitetura do INSIGMA serve como base para a arquitetura do sistema de notificações do COISA. Porém, para se adaptar ao problema específico no contexto

do LARIISA, algumas tecnologias foram substituídas àquelas apresentadas no INSIGMA. Um exemplo é que, ao invés de utilizar desenvolvimento de *webservices* com SOAP, foi utilizado *Representaional State Transfer* (REST) para a oferta de serviços.

2.6.4. InteliMED

Menezes et al. (2011) descreveram a proposta do Sistema Móvel de Apoio a Decisão Médica Aplicado ao Diagnóstico da Asma (InteliMED). O foco do projeto foi desenvolver um sistema de suporte remoto de diagnóstico médico inicial, utilizando tecnologias inteligentes em dispositivos móveis, auxiliando assim o processo de atenção básica à saúde. Esse trabalho foi escolhido por utilizar técnicas de apoio à decisão em um sistema de saúde com dispositivos móveis, como ocorre no projeto COISA.

Para o processo de desenvolvimento de software, a metodologia adotada foi uma mescla do *Scrum* (SCRUM, 2014) com a XP (*eXtremming Programming*) (XP, 2014) (MENEZES E GUSMÃO, 2013). O projeto foi desenvolvido para a plataforma Android, versão 2.2. Portanto, a linguagem de programação utilizada foi a Java.

O Jenkins (JENKINS, 2014) foi utilizado para aplicar integração contínua no projeto, juntamente com o *plugin* do Sonar (SONAR, 2014), voltado para verificação da qualidade de código e boas práticas de programação. Os testes unitários foram desenvolvidos utilizando o *framework JUnit* para Android (JUNITANDROID, 2014). Finalmente, para versionamento de código foi adotado o SVN (SUBVERSION, 2014).

2.6.4.1. Arquitetura do InteliMED

A arquitetura do InteliMED divide o sistema em dois módulos: Aplicação Móvel e Servidor (MENEZES E GUSMÃO, 2013). A figura 14 ilustra esse modelo.

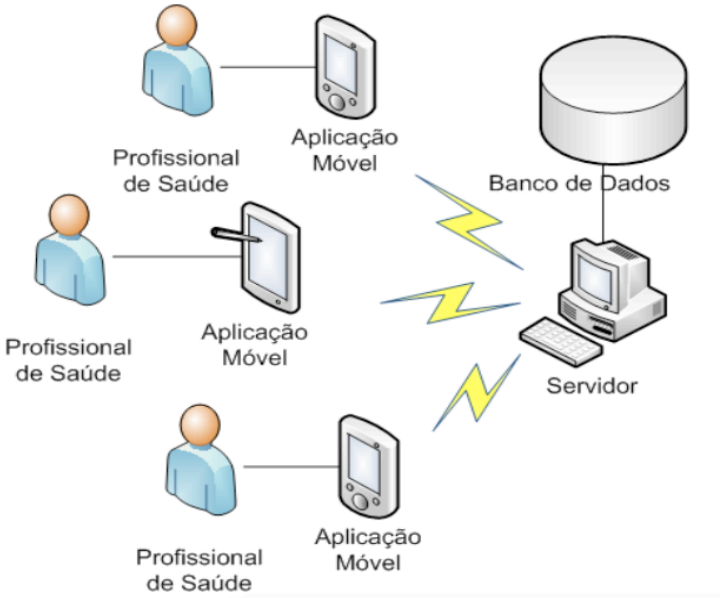

Figura 14 - Arquitetura do InteliMED (MENEZES E GUSMÃO, 2013)

No InteliMed, a regra de negócio principal é prover o diagnóstico de asma. Essa regra foi implementada de modo que o usuário responsável por fornecer o diagnóstico consiga usar o aplicativo independente da conexão de dados, possibilitando o uso da aplicação em ambientes onde o acesso à internet é limitado ou até mesmo inexistente.

Para auxiliar a tomada de decisão no ato do diagnóstico, utilizou-se mineração de dados com a técnica de árvore de decisão. Assim, quando o médico responde um questionário, coletando evidências, a árvore de decisão é percorrida de acordo com as respostas, processando e gerando o resultado e informando ao usuário. Logo após o resultado, é solicitada a confirmação do médico para validar a decisão, possibilitando também a inclusão de comentários informando o motivo de não corroborar com a decisão do aplicativo ou para destacar informações pertinentes à análise ou auditoria do diagnóstico.

As respostas são armazenadas primeiramente na base de dados do dispositivo através do Sistema Gerenciador de Banco de Dados *SQLite*. Quando a conexão com a internet é reestabelecida, os dados são enviados ao Servidor, concluindo a sincronização.

Além da funcionalidade apresentada anteriormente, existem mais duas de igual importância: a primeira é "Enviar evidência", responsável por remeter informações do diagnóstico, armazenadas localmente no dispositivo, para o servidor;

"Atualizar árvore" é a segunda funcionalidade. Tem como objetivo possibilitar a atualização da árvore de decisão do dispositivo.

O servidor, além de armazenar a árvore de decisão para o diagnóstico da asma, deve ser implantado em uma Unidade de Saúde Básica, fazendo com que um *script* inteligente espelhe a comunidade assistida pela Unidade de Saúde. A figura 15 apresenta tela do sistema servidor InteliMED, com visualização da árvore de decisão.

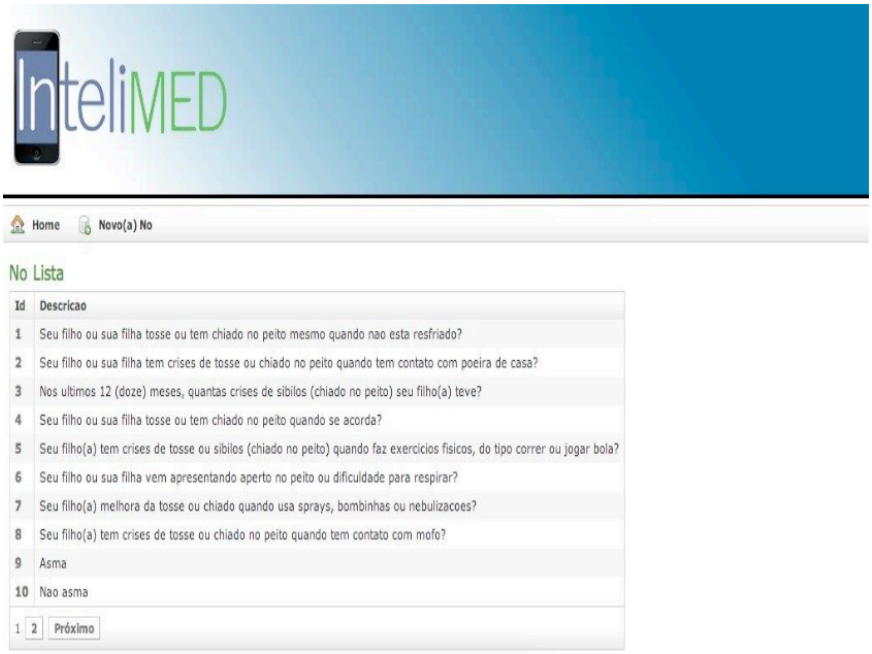

Figura 15 - Arquitetura do InteliMED (MENEZES E GUSMÃO, 2013)

2.6.4.2. Arquitetura dos módulos de Aplicação Móvel e Servidor

A arquitetura da aplicação móvel possui três componentes principais:

- **Database**: armazena árvore de decisão e evidências coletadas por meio dos diagnósticos;
- **Diagnostic**: exibe o questionário, percorre a árvore de decisão, exibe o diagnóstico e coleta evidências da opinião médica;
- **Communication**: responsável pela troca de mensagens entre o aplicativo cliente e o servidor.

A figura 16 reflete o modelo arquitetural com os três componentes listados acima:

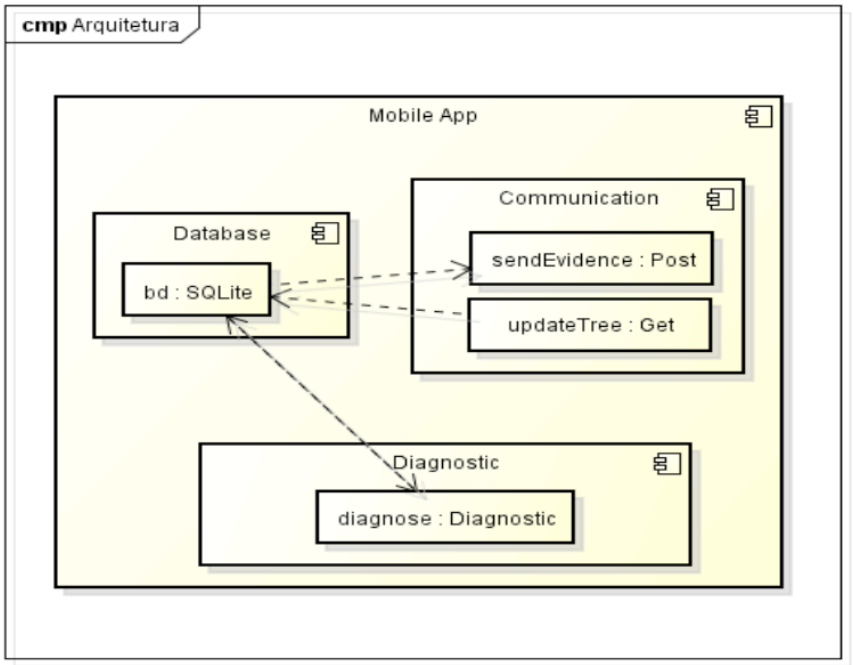

Figura 16 - Arquitetura da aplicação Móvel do InteliMED (MENEZES E GUSMÃO, 2013)

A arquitetura do servidor InteliMED tem dois componentes:

- *Controller*: trata requisições advindas do dispositivo móvel ou da interface web;
- *Domain*: mantém as entidades do sistema, definindo a estrutura de dados da árvore de decisão.

A figura 17 a seguir revela a arquitetura básica do servidor.

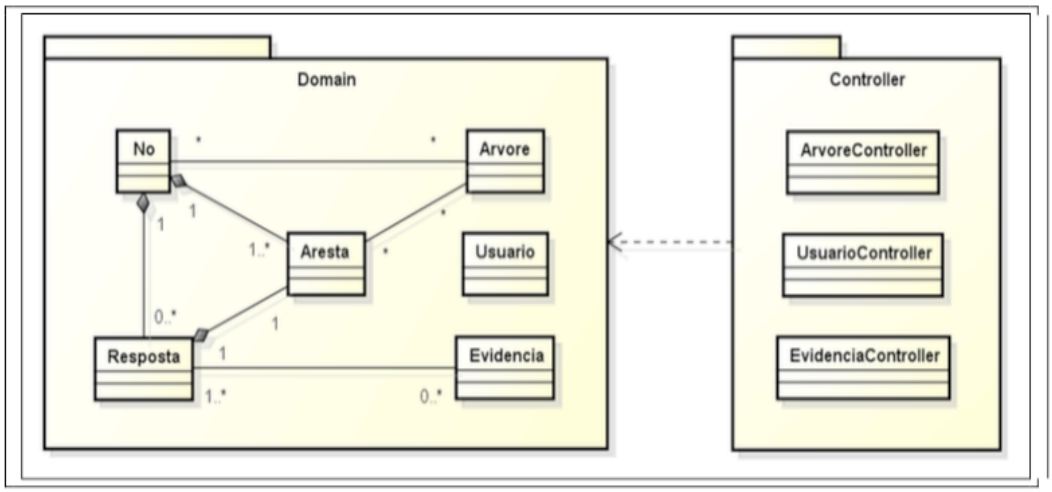

Figura 17 - Arquitetura do servidor do InteliMED (MENEZES E GUSMÃO, 2013)

2.6.4.3. Mineração de dados

A mineração de dados é um mecanismo que permite a descoberta de padrões consistentes a partir de grandes bases de dados. No InteliMed, o algoritmo escolhido para processar a mineração de dados foi a árvore de decisão. A ferramenta de extração de conhecimento foi a WEKA (Waikato *Environment for Knowledge Analysis*), a qual mantém diversos algoritmos para mineração de dados (HALL et al., 2009).

Uma equipe de saúde selecionou um questionário com 40 questões que serviram como entrada para o algoritmo da mineração de dados no InteliMed. Um total de 93 pacientes responderam o questionário. Para aumentar o espaço amostral, a equipe técnica de saúde simulou mais 50 cenários, totalizando 143 instâncias, considerado um número pequeno mas suficiente para gerar uma taxa de acerto de 91,61% com o algoritmo de árvore. Essa taxa é considerada aceitável.

Em todos os experimentos foi utilizado o algoritmo J48, uma implementação da técnica para construção de árvore de decisão conhecida como C4.5. A seguir, as variações utilizadas nos experimentos:

- V1: Poda – algoritmo executado com a opção de poda ativa;
- V2: Sem poda – algoritmo executado com a opção de poda desativada;
- V3: Pré-processamento 1 – seleção de atributos utilizando o algoritmo *geneticSearch* antes da execução do algoritmo de árvore de decisão;

2.6.5. O COISA e os trabalhos relacionados

Para consolidar a relação dos trabalhos relacionados com o projeto COISA, foram destacados os seguintes pontos para a comparação dos sistemas:

1. O sistema é sensível ao contexto;
2. O sistema é voltado para a área de saúde;
3. O sistema possui integração com o SINAN;
4. Utilização de RES com o *openEHR*;
5. Método para tomada de decisão ou inferência;
6. Ontologia como método para tomada de decisão;
7. Facilidade para importação/exportação de dados

A tabela 01 apresenta a comparação dos sistemas estudados.

Sistema	1	2	3	4	5	6	7
Sisa	+	+	-	-	+	-	-
CLARIISA	+	+	-	-	+	+	-
INSIGMA	+	-	-	-	+	+	-
InteliMED	+	+	-	-	+	-	-
COISA (Trabalho proposto)	**+**	**+**	**+**	**+**	**+**	**+**	**+**

Tabela 01 - Comparação entre os trabalhos relacionados e o trabalho proposto

2.7. Considerações finais do capítulo

Este capítulo descreveu o referencial teórico sobre contexto, geolocalização, ontologias, projeto LARIISA, Registros Eletrônicos de Saúde e SINAN. Tais conceitos fornecem a base teórica para a especificação e implementação do projeto COISA (capítulos 03 e 04).

Os trabalhos relacionados trouxeram soluções envolvendo situações e tecnologias semelhantes às utilizadas no projeto COISA. Cada projeto estudado teve sua relevância e influência na elaboração e implementação do sistema COISA.

O capítulo a seguir possui todo o conhecimento funcional do projeto COISA, envolvendo análise de requisitos e modelagem de sua arquitetura como uma extensão do projeto LARIISA.

3. PROJETO COISA

Este capítulo apresenta o projeto COISA: Conselheiro Inteligente de Saúde da Plataforma LARIISA. Trata-se de um trabalho de modelagem de um sistema que estende a arquitetura do LARIISA para incluir tecnologias que provêem suporte ao monitoramento e alertas sobre doenças na sua estrutura.

Para encaixar o domínio de monitoramento e alertas no modelo do LARIISA, foram elaboradas duas Interfaces Gráficas de Usuário (GUI). A primeira tem o papel de coletar dados por meio de aplicações web e *mobile* para cadastro de entidades e regras. A segunda interface é responsável por processar e publicar os dados. Um exemplo de publicação seria informar o usuário sobre um risco de doença à sua saúde.

O monitoramento de doenças é aplicado na parte central do sistema, localizada entre as duas interfaces, e baseado em regras, definidas por meio de ontologias ou pelo padrão *openEHR*, que auxiliam a tomada de decisão.

O cliente desenvolvido para dispositivos móveis é chamado COISA Mobile (COBILE). Já a aplicação do lado servidor é o *Geographic Health System* (GeoHS), responsável pelo processamento dos dados.

As Interfaces Gráficas de Usuário (GUIs) do COBILE e do GeoHS representam o Provedor de Contexto do LARIISA, que lida com a coleta de dados dos usuários via dispositivo móvel ou via sistema web. Além disso, possuem capacidade de enviar informações para o servidor com o contexto específico para uma determinada situação.

Os arquétipos dos Registros Eletrônicos de Saúde, as ontologias utilizadas para inferência e a aplicação do monitoramento de doenças são executados pelo *core* (núcleo do sistema) do GeoHS. Este fica responsável por receber os dados de entrada do sistema coletados mediante a GUI e processar os dados de modo inteligente, gerando contexto através do LARIISA e fabricando alertas e relatórios para usuários comuns e à governança de saúde.

A figura 18, a seguir, explana este cenário. Um usuário insere dados através do GeoHS ou do COBILE. O LARIISA, então, define o contexto com as restrições dos arquétipos e realiza inferência com base nas ontologias ou no *openEHR*, gerando o retorno desejado pelo usuário.

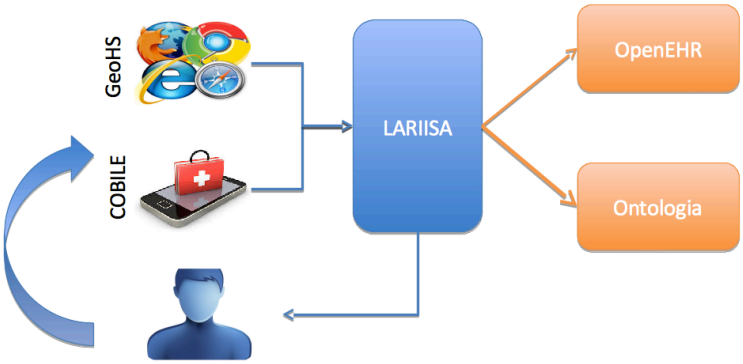

Figura 18 – Modelo Arquitetural do COISA

Além da visão sistêmica mostrada na figura 18, uma classificação de papéis se faz necessária para se ter uma análise com visão de um analista de negócio. Para isso, a figura 19 ilustra uma solução do projeto em alto nível, mais facilmente entendido por um usuário aquém do conhecimento técnico. Nota-se, no lado esquerdo da figura, que existem dois tipos de interessados no sistema: produtores e consumidores.

Os produtores são os usuários responsáveis por injetarem informações no sistema, formando a base de dados, os arquétipos e o repositório de conhecimento. Os consumidores são os interessados em usufruir dos dados já informados pelos produtores e que já estejam armazenados nas bases do sistema.

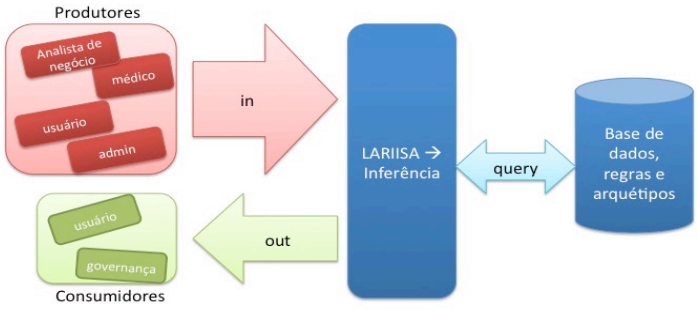

Figura 19 - Visão de papéis e fluxo básico de dados do COISA

A seguir, é apresentada uma lista com a definição de cada papel existente nos produtores e nos consumidores:

- **Produtores**
  - **médico:**
    - **formação de histórico de antecedentes do paciente**. Assim que o médico diagnosticar uma doença, O sistema irá associar essa doença ao paciente, formando o seu histórico epidemiológico ao preencher o Registro Eletrônico de Saúde através da implementação do *openEHR*.
  - **analista de negócio**:
    - **alimentar regras do sistema**. O analista de negócio precisa se reunir com especialistas da área de saúde para definir as regras de negócio que irão estruturar a lógica do sistema. Essas regras fornecem informações para que o LARIISA realize inferências necessárias para responder requisições. Exemplo de regra: Se uma pessoa não foi vacinada contra a malária, deve ser informada que necessita ser vacinada caso queira ir a uma determinada região que tem foco recente de malária. Estas regras devem ser especificadas no modelo de arquétipos e templates do *openEHR*.
  - **agente de saúde**: responsável por incluir eventos de acompanhamento dos pacientes. No caso de um evento grave, envia uma notificação de

urgência ao hospital. O aplicativo é responsável por acionar a unidade mais próxima.

- **admin**: O usuário administrador do sistema (*admin*) é responsável por **realizar as configurações iniciais do sistema**, como importar doenças existentes da CID-10[3], por exemplo. Além disso, enquanto o SINAN não possui *webservice* capaz de fornecer acesso aos dados dos agravos, o usuário *admin* é o responsável pela **importação de agravos vindos do SINAN**, mediante arquivo de extensão ".dbf" (usada no Sistemas de Gerenciamento de Banco de Dados dBASE) extraído do SINAN.

- **Consumidores**
  - **usuário**: o usuário como consumidor utiliza um aplicativo de dispositivo móvel com duas funcionalidades básicas:
    - Receber alertas sobre focos de doenças em tempo real, à medida que se desloca de uma região a outra.
    - Consultar um destino para verificar quais riscos existem para sua saúde na região escolhida.
  - **governança**: a governança no COISA é o papel representado pela administração de um posto público de saúde, de uma secretaria de saúde ou de representantes do próprio ministério de saúde, que podem ser compostos por diretores, gestores, administradores, secretários e ministros de saúde. Usuários consumidores com o papel de governança têm acesso a informações privilegiadas, como por exemplo verificar em um período qual a doença que mais ocorreu em uma determinada região. O governo pode usar esta funcionalidade para o controle de endemias, por exemplo.

Além dos papéis apresentados, a figura 19 mostra duas setas, indicando o sentido que os dados transitam. A seta *in* mostra os dados sendo enviados pelos produtores para dentro do *core* do sistema. A seta *out* mostra o sentido dos dados saindo do *core* e indo como resposta a solicitações dos consumidores.

[3] A Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde (CID-10) tem a função de padronizar e catalogar doenças e problemas de saúde (BRASIL, 2008). É por meio da CID-10 que programas e sistemas podem referenciar classificações de doenças.

Apesar da visão macro do sistema ter sido apresentada anteriormente, é necessária também uma representação mais formal para ajudar no desenvolvimento do sistema. É proposta, então, uma especificação do COISA por meio das descrições dos requisitos funcionais do sistema, diagramas de caso de uso e diagramas de classe, explicitando as funcionalidades e os relacionamento das entidades que o sistema necessita para o cumprimento do projeto.

A próxima seção descreve os requisitos funcionais do COISA, exemplificando as principais funcionalidades dos produtores e consumidores do sistema.

3.1. Requisitos Funcionais

Nesta seção são descritos os requisitos funcionais necessários para o funcionamento do sistema. divididos entre produtores (entrada de dados), consumidores (saída de dados) e requisitos gerais:

- **Requisitos de produtores**
  - Cadastro de doenças
  - Cadastro de foco de doenças
  - Cadastro de regras para alertar usuários
  - Criação de *templates* e arquétipos do openEHR para modelagem clínica.
- **Requisitos de consumidores**
  - Usuário escolhe um local como destino e recebe conselho do aplicativo sobre a região escolhida
  - Usuário recebe notificações do sistema ao entrar ou sair de uma área demarcada
  - Usuário capaz de ver mapas epidemiológicos
- **Requisitos Gerais**
  - Login de usuário

3.1.1. Cadastro de doenças

O usuário com o papel de administrador do sistema poderá cadastrar doenças por meio da importação de doenças já existentes na CID-10, afim de facilitar o cadastro de doenças e também pela confiabilidade das informações contidas na CID-10.

O módulo Web do COISA tem uma tela específica para importação de doenças, que só será exibida para o usuário com o papel de administrador. O controle de acesso, que é descrito com mais detalhes no capítulo 04, deverá restringir esta tela para que apenas o administrador tenha acesso.

O cadastro de doenças é dependente de algumas entidades, além da entidade que representa a doença. Uma delas é a entidade *Sintoma*. Um sintoma caracteriza um efeito causado pela doença. Portanto, são necessárias duas entidades para que a doença possa ser armazenada no sistema: doença e sintoma.

A tabela 02 a seguir define a entidade **sintoma**:

Sintoma			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Descrição	Descrição do sintoma.	Texto com 100 caracteres	Sim
Cronologia	Identificação dos aspectos relacionados ao tempo e sequência de evolução dos sintomas.	Texto com 100 caracteres	Não
Localização Corporal	Local dos sintomas, irradiação e profundidade.	Texto com 100 caracteres	Não
Qualidade	Descrição que o paciente faz de suas percepções.	Texto com 100 caracteres	Não
Quantidade	Intensidade, frequência, número de vezes em que o fenômeno ocorreu, intervalo entre os episódios, volumes de secreções, abaulamentos, edemas	Texto com 200 caracteres	Não
Circunstâncias	Local, atividade que exerce no momento da ocorrência do sintoma, exposição a fatores ambientais, ingestão de alimentos.	Texto com 100 caracteres	Não
Manifestações Associadas	Ocorrência de outros sintomas em decorrência deste sintoma.	Texto com 100 caracteres	Não
Doenças	Lista de doenças em que este sintoma ocorre.	Lista com zero ou mais doenças	Não

Tabela 02 - Entidade que representa um Sintoma

A tabela 03 a seguir descreve a entidade **doença**:

Doença			
Atributo	**Descrição**	**Metadado**	**Mandatório**
cid	Código CID10 da doença	Texto com 20 caracteres	Sim
Nome	O nome Popular pelo qual a doença é conhecida.	Texto com 50 caracteres	Sim
Nome Cientifico	O nome científico da doença em questão.	Texto com 50 caracteres	Não
Descrição	Descrição da doença	Texto com 200 caracteres	Não
Agente Causador	Agente causador da doença, ou seja o que provoca a doença em questão (As opções são: Protozoários, Bactérias, fungos e vírus).	Enumeração	Não
Sintomas	O conjunto dos sintomas que se manifestam para aquela doença.	Lista com zero ou mais sintomas	Não
Tempo de Expiração	Tempo de expiração do foco. Após o foco ser registrado, é contado o tempo de expiração. Depois que o tempo é expirado, a área demarcada associada ao foco da doença também é expirada, não podendo mais ser mostrada pelo aplicativo	Um número natural para representar os dias.	Sim
Profilaxia	Conjunto de profilaxias de uma doença. Profilaxia determina um tipo de prevenção da doença.	Texto com 200 caracteres	Não

Tabela 03 - Entidade que representa uma doença

3.1.2. Cadastro de foco de doenças

O sistema deve disponibilizar a importação dos focos de doenças do SINAN. O papel responsável pela importação de foco de doenças é o de administrador. A importação dos agravos e demais focos é necessária para a confiabilidade dos dados. O SINAN é o local designado pelo Ministério da Saúde para armazenar todos os casos de doenças de notificação compulsória, exatamente o que o COISA precisa para produzir informações confiáveis para a população.

O médico realiza a consulta com o paciente, colhendo dados necessários para avaliar se realmente há suspeita ou confirmação de doença. Se for confirmada a suspeita ou a confirmação do agravo, o médico preenche as fichas FIN ou FII, como detalhado no capítulo 02 (página 38, item 2.5) e em (BRASIL, 2011).

Logo após o preenchimento da ficha, o posto de saúde responsável pela notificação deve enviar os dados para o SINAN em um prazo máximo de 15 dias. Assim que os dados estiverem no SINAN, qualquer usuário com o perfil de administrador do COISA é capaz de realizar a importação dos agravos.

Na importação, o COISA recebe o arquivo de extensão ".dbf" do administrador do sistema e realiza o processamento, conseguindo os dados da doença e do endereço, no caso o bairro, do foco da doença, e salvando o registro de cada agravo no banco de dados.

Em conjunto com a localização e a doença, o cadastro também contém a data da ocorrência da doença. Deste modo, criar um relatório de doenças ocorridas em uma região, em um determinado período, ficará viável de ser implementado.

Para a realização da importação de focos de doença pelo administrador, foram criadas as entidades *Ponto* e *FocoDoenca*, apresentadas nas tabelas a seguir com os atributos necessários para esta importação:

Ponto			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Latitude	Latitude da localização	Número real. Utilizar precisão do ponto do GPS da API de localização do google.	Sim
Longitude	Longitude da localização	Número real. Utilizar precisão do ponto do GPS da API de localização do google.	Sim

Tabela 04 - Representação da entidade *Ponto*

FocoDoenca			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Ponto	Ponto geográfico da localização	Ponto.	Sim
Doença	Doença do paciente.	Doença	Sim
Data	Data e hora em que ocorreu o registro do foco no sistema.	Data	Sim

Tabela 05 - Entidade que representa um foco de doença

3.1.3. Cadastro de regras para alertar usuários

Para que um usuário consumidor seja alertado sobre determinado risco para a sua saúde, são necessárias regras especificando quais premissas devem ser satisfeitas para o disparo de um alarme. Por exemplo: se um usuário nunca teve catapora e deseja ir para um local com agravos de catapora, este usuário deve ser

alertado. Este exemplo é bem simples, mas deve ser considerado como teste de aceitação para o desenvolvimento do sistema.

Para a criação de regras, é necessário realizar a modelagem de *templates* e arquétipos do *openEHR*. Assim, um modelo clínico confiável é desenvolvido independente das ferramentas e conhecimentos específicos na área da informática. O arquétipo contém todas as informações pertinentes do ponto de vista médico, fazendo com que o COISA tenha uma base confiável de regras validadas por médicos de todo o mundo.

Ao realizar a modelagem dos *templates* e dos arquétipos, o COISA poderá usar a ontologia para classificar o risco de saúde do usuário em uma determinada região. Dependendo da classificação de saída da ontologia, é emitido um sinal de alerta ao usuário. Esse processo é necessário para a implementação do próximo requisito funcional: Usuário Consumidor recebe notificações do sistema ao entrar ou sair de uma área demarcada (ou área de risco).

3.1.4. Cadastro de doenças que compõem perfil epidemiológico do usuário

Este é o último requisito dos produtores, que define que o médico especialista pode realizar a tarefa de cadastrar doenças para formar o perfil epidemiológico de um determinado paciente, agregando o diagnóstico do paciente a seu Registro Eletrônico de Saúde (RES). O RES do paciente (usuário comum) é composto pelas doenças (patologias) obtidas durante toda a vida deste paciente, somadas ao cronograma de vacinas, dados cadastrais, entre outros.

Somente o médico é capaz de cadastrar doenças para manter o histórico epidemiológico dos pacientes, atualizando o RES quando necessário. O paciente obrigatoriamente deverá ter o seu cadastro único de saúde (cadastro no SUS) para que o médico possa cadastrar as ocorrências no RES.

No caso de o paciente estar se aproximando de uma região que contém um foco de malária, por exemplo, o sistema pode alertar o paciente, informando que é necessário tomar a vacina contra a malária, dependendo da condição do seu RES.

Os Registros Eletrônicos de Saúde dos pacientes são essenciais para realização de inferências sobre os pontos de interesse de cada usuário consumidor

(paciente). Os pontos de interesse dos usuários se adequam exatamente ao perfil individual de cada usuário.

Como exemplo, pode-se considerar um usuário que porta seu dispositivo móvel com o aplicativo COBILE instalado, e nunca fora vacinado contra a malária. O COISA possui o RES desse usuário e, portanto, tem conhecimento de que o usuário precisa de um alerta sobre o risco apresentado ao chegar em uma região próxima de onde houve registro de Malária e esse registro ainda é válido, ou seja, o risco ainda é pertinente.

No COISA, o risco dos agravos de doenças para os usuários é considerado apenas se o tempo de expiração da doença ainda for válido. Sendo assim, no exemplo anterior, se a malária foi detectada em Janeiro de 2014, ela não deve ser mais preocupante para o usuário no ano de 2015, pois não há mais ameaça deste foco para os usuários consumidores. Uma doença expirada só tem interesse para a governança na área de saúde, que pode emitir relatórios de doenças por períodos ou verificar gráficos estatísticos.

3.1.5. Usuário escolhe um local como destino e recebe conselho do aplicativo sobre a região escolhida

Os requisitos funcionais desse tópico são os que envolvem integração efetiva direta com o LARIISA, devido ao requerimento de representação do conhecimento do RES do paciente e do perfil epidemiológico de uma determinada região.

Para representação do conhecimento sobre o RES de um usuário do sistema, é necessário criar uma ontologia com o domínio baseado no mapeamento do usuário com o seu RES. A plataforma LARIISA precisa dessa ontologia para responder a solicitações de usuários que desejam viajar para uma determinada cidade, estado, ou país, ou que apenas necessitam obter informações sobre determinada região.

Um perfil epidemiológico de uma região abrange todas as doenças que pertencem a essa região. Portanto, dado que o sistema conhece o perfil epidemiológico do usuário, através de seu RES, e se este usuário deseja viajar para uma determinada cidade ou país, o sistema deve informar os riscos que podem afetar a saúde do usuário, caso o usuário se aproxime àquela determinada região ou

deseje chegar naquela região, colocando tal região como destino no aplicativo desenvolvido para o seu dispositivo móvel.

Dependendo do perfil epidemiológico do usuário, o perfil epidemiológico de uma região poderá servir como base de alerta para este usuário. Exemplo de um cenário: Um determinado usuário do sistema mora no Ceará e deseja visitar alguns familiares que moram em uma região do Amazonas.

O sistema tem o perfil epidemiológico do usuário e sabe que o usuário nunca foi vacinado contra a malária. O sistema também conhece o perfil epidemiológico da região escolhida no Amazonas e sabe que existem agravos de malária na região. Portanto, o sistema deve avisar ao usuário para se vacinar contra a malária antes de realizar a viagem desejada.

Os requisitos funcionais que auxiliam a decisão sobre qual situação um usuário deve ser notificado, dependendo do perfil epidemiológico da região, devem ser negociadas com um especialista em saúde. O profissional que mais se adequa ao perfil de definição dessas regras de negócio é o analista de negócio, juntamente com validações de um profissional da área de saúde (um médico, por exemplo). Portanto, ao perfil *analista de negócio* é designado o papel de criar arquétipos e *templates* do *openEHR*, compondo, assim, a modelagem clínica.

O sistema COISA possui as ferramentas prontas para serem alimentadas com regras de negócio definidas pelos médicos ou especialistas em endemias, como sugere o requisito funcional anterior (*Cadastro de regras para alertar usuários*).

O escopo deste trabalho não abrange o cadastro de muitas regras de negócio, e sim a automatização do processo para que, futuramente, seja possível especialistas de saúde interessados negociarem com especialistas de Tecnologia da Informação e criar regras no *openEHR* de acordo com as especificidades do COISA.

3.1.6. Usuário recebe notificações do sistema ao entrar ou sair de uma área de risco

Como um consumidor do COISA, o usuário comum está apto a receber notificações do sistema por intermédio de um dispositivo móvel. No dispositivo é preciso instalar o aplicativo do LARIISA responsável pelas notificações, o COBILE. O aplicativo consegue sentir o ambiente em que o usuário consumidor está

localizado. Se o usuário entrar em alguma área de risco à sua saúde, o aplicativo envia uma notificação para que o usuário tenha conhecimento de tal ameaça.

O usuário consumidor será notificado quando entrar ou sair de uma determinada área de risco de interesse do usuário. O interesse do usuário é baseado no seu RES. A área de risco é definida pelas coordenadas coletadas pelo sistema durante a importação dos focos de doenças, e pelo raio da área, grande o suficiente para o usuário manter a distância necessário do foco.

O sistema pode ter mais de uma área de risco ativa para cada usuário, dependendo dos interesses epidemiológicos deste usuário. Um exemplo de várias áreas de ameaça representadas em um mapa é demonstrada na seguinte figura:

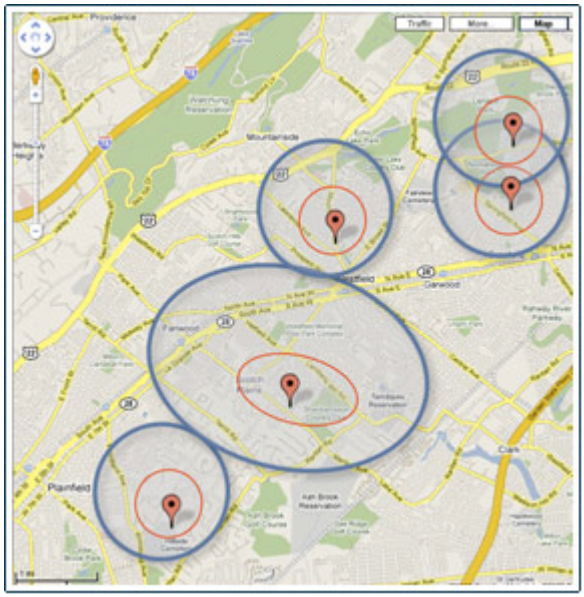

Figura 20 - Várias áreas demarcadas em um mapa (MOBILETIME)

Além do ponto geográfico (latitude e longitude) e do raio, também é definido um tempo para a área de risco ser válida. Este tempo é configurado de acordo com o arquétipo definido para uma doença específica. O responsável da área de saúde editará o tempo de cada doença no arquétipo. Assim, o sistema notifica o usuário apenas se a área de risco estiver ativa e for pertinente, de acordo com o RES do usuário. Segue a tabela 06, que define a entidade para uma área de risco.

AreaRisco			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Ponto	Ponto geográfico da	Ponto.	Sim

	localização		
Raio	Tamanho do raio da distância da localização para o ponto de interesse.	Número real.	Sim
Doença	Doença do paciente.	Doença	Sim

Tabela 06 - Entidade que representa uma área de risco

3.1.7. Geração de mapa epidemiológico

O usuário consumidor pode ter a opção de ver as doenças não expiradas registradas através de um mapa. Para isso, é solicitado que o usuário entre com uma doença na interface do aplicativo. Após escolher a doença, um mapa da região onde o usuário está localizado é apresentado, contendo os focos da doença válida (não expirada) escolhida pelo usuário.

3.1.8. Login de usuário

O sistema deverá ter uma tela de login para que o usuário consiga se conectar ao sistema e realizar as operações possíveis de acordo com o seu perfil. É preciso ter duas entidades para que este requisito seja satisfeito. São elas: Usuário e Permissão. As tabelas 07 e 08 representam estas entidades.

Usuário			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Login	Identificador de login do usuário.	Texto com 50 caracteres	Sim
Nome	Nome do usuário.	Texto com 50 caracteres	Sim
Sobrenome	Sobrenome o usuário.	Texto com 50 caracteres	Sim
Senha	Senha de login.	Texto com 200 caracteres	Sim
Email	Email do usuário.	Texto com 50 caracteres	Sim
Data de Cadastro	Data de registro do usuário.	Data	Sim
Permissões	Lista de permissões do usuário.	Lista com zero ou mais Permissões	Não
Doenças	Lista de doenças do usuário.	Lisita com zero ou mais Doenças	Não

Tabela 07 - Entidade que representa um usuário

Permissão			
Atributo	**Descrição**	**Metadado**	**Mandatório**
Descrição	Descrição da permissão.	Texto com 500 caracteres	Sim
Usuários	Lista de usuários.	Lista com zero ou mais Usuários	Sim

Tabela 08 - Entidade que representa uma permissão

3.2. Diagrama de Caso de Uso

Nesta seção, os casos de uso representam o comportamento dos atores de sistema: admin, médico, analista de negócio e usuário comum.

As figuras 21, 22, 23 e 24 a seguir mostram os casos de uso principais do COISA.

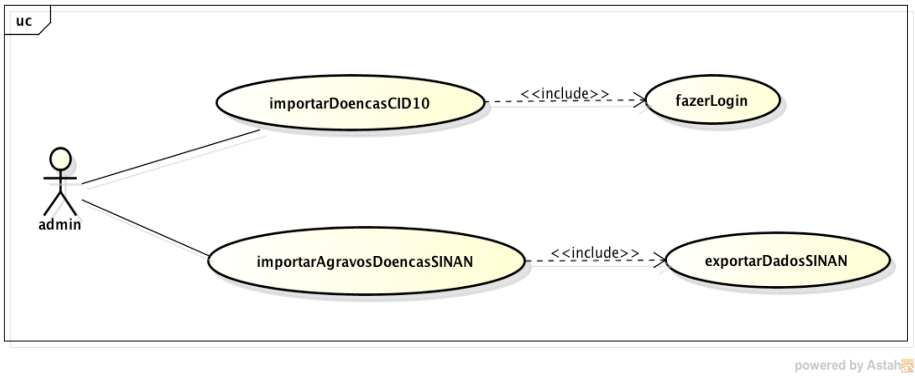

Figura 21 - Caso de uso do ator *admin*

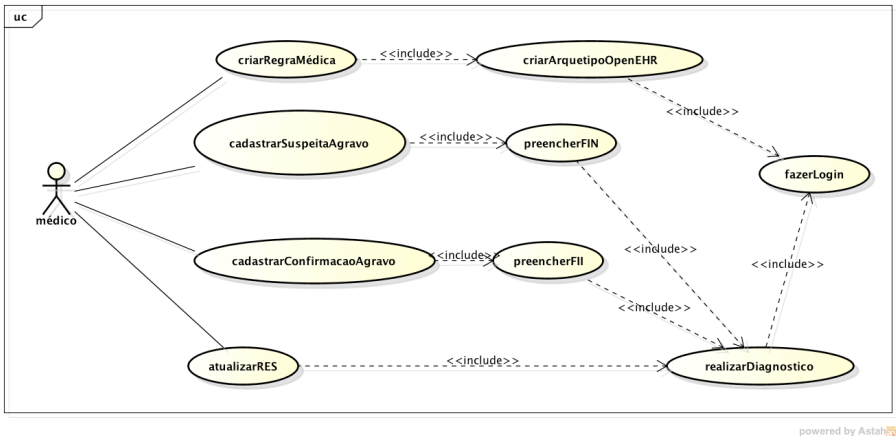

Figura 22 - Caso de uso do ator *médico especialista*

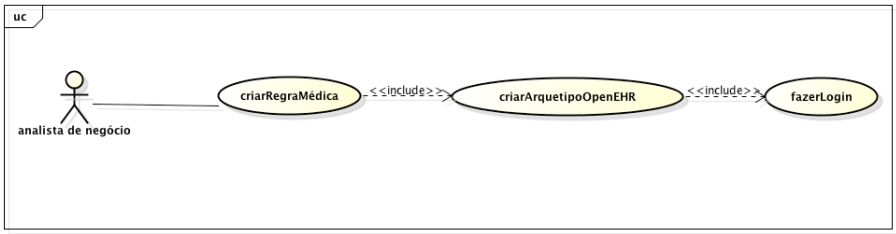

Figura 23 – Caso de uso do analista de negócio

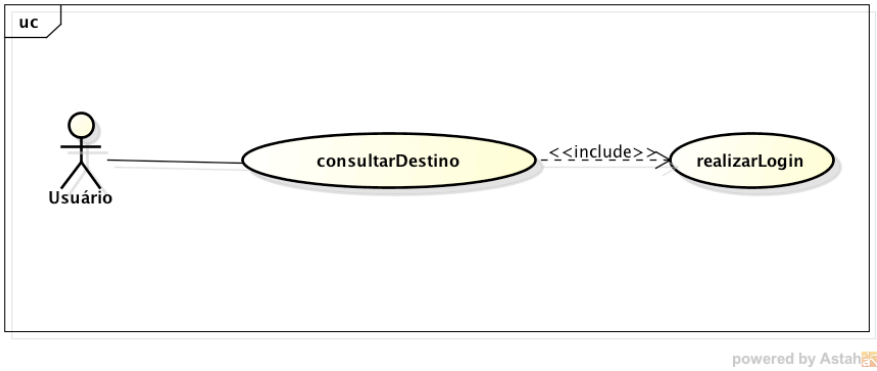

Figura 24 - Caso de uso do ator *usuário*

3.3. Diagrama de Classes

A figura 25 representa o diagrama de classes de apenas um pacote Java, o pacote de modelo (no código este pacote é nomeado de *model*), responsável por definir as entidades que representam as classes básicas para o funcionamento do COISA.

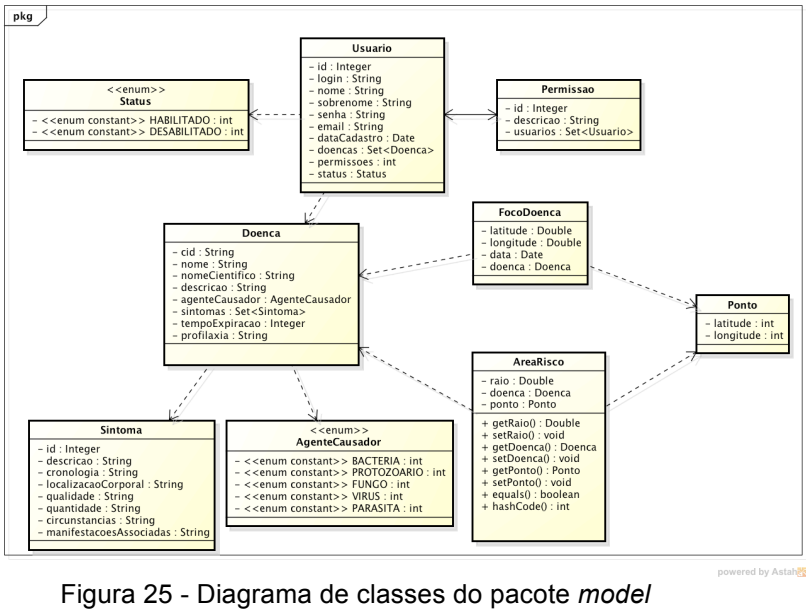

Figura 25 - Diagrama de classes do pacote *model*

3.4. Considerações Finais do Capítulo

Este capítulo explicou todo o negócio envolvido do COISA, mostrando detalhadamente as regras funcionais necessárias, além dos diagramas de caso de uso e diagrama de classe, para que os sistemas GeoHS e COBILE fossem implementados.

Para o entendimento dos requisitos funcionais, foram definidos os papéis existentes no sistema e quais as responsabilidades desses papéis. Logo após, cada requisito fez referência a um ou mais papéis, descrevendo com mais detalhe as suas funções.

As entidades necessárias para a modelagem do sistema foram descritas utilizando o formato de tabelas, detalhando os campos (atributos) necessários de cada entidade e o relacionamento das entidades entre si. Isso facilitou tanto a modelagem de dados do sistema, quanto a modelagem de entidades no código Java.

No capítulo seguinte (cap. 04) são discutidos os aspectos de implementação dos sistemas do projeto COISA.

4. IMPLEMENTAÇÃO DO PROJETO COISA

Este capítulo aborda os aspectos detalhados de implementação dos sistemas GeoHS e COBILE. São definidos os requisitos *não funcionais*, a arquitetura de software na qual são desenvolvidos, as tecnologias principais utilizadas no desenvolvimento e na implantação do modelo de testes automatizados e a metodologia escolhida para o desenvolvimento de software.

Algumas ferramentas e tecnologias foram estudadas para que o COISA pudesse ser desenvolvido com o menor custo possível, obtendo menos tempo de desenvolvimento e melhor documentação para posterior continuação do projeto. As ferramentas estudadas durante a criação deste documento de dissertação servirão como base para a implementação dos requisitos funcionais que estão descritos nos próximos tópicos deste capítulo.

4.1. Metodologia de desenvolvimento de software

A metodologia Ágil (AGILE, 2015) foi escolhida para o desenvolvimento do COISA devido sua flexibilidade aos requisitos do cliente. Como o sistema tende a crescer posteriormente, a metodologia ágil permite uma forma iterativa e não rígida para que as necessidades do cliente se adeque a cada iteração.

O desenvolvimento ágil de software também tem a característica de minimizar os riscos, devido ao curto período de tempo que é desenvolvido uma pequena parte do software e apresentado ao cliente. A cada iteração, é entregue software funcional para o cliente, sempre mantendo a simplicidade e o foco no negócio.

Se o cliente requisita mudança, os planos de projeto são mudados e o planejamento antigo deve ser remodelado. É priorizada uma colaboração com os clientes, ao invés de contratos de negócio.

4.2. GeoHS

Como foi apresentado no capítulo 03, o GeoHS é a aplicação web do COISA, responsável pelos papéis seguintes:

- **Entrada de dados**: páginas web com telas de cadastro de entidades (doenças, usuários, sintomas, etc);
- **Processamento de dados**: Realiza inferência e processamento de dados, gerando saída para usuário, caso for necessário;
- **Saída de dados**: interface web para usuário poder verificar os dados, lista de doenças, lista de usuários, relatórios, etc.

4.2.1. Arquitetura de Software do GeoHS

O GeoHS foi desenvolvido na plataforma Java, facilitando a implantação em qualquer sistema operacional devido a sua interoperabilidade. A versão utilizada do Java foi a 8, mais recente versão no momento da escrita deste trabalho.

O modelo arquitetural adotado foi o padrão *Model View Controller* (MVC) por meio do *JavaServer Faces* (JSF). A figura 26, a seguir, mostra as diversas camadas do sistema incluídas no pacote "br.com.dazen.geohs".

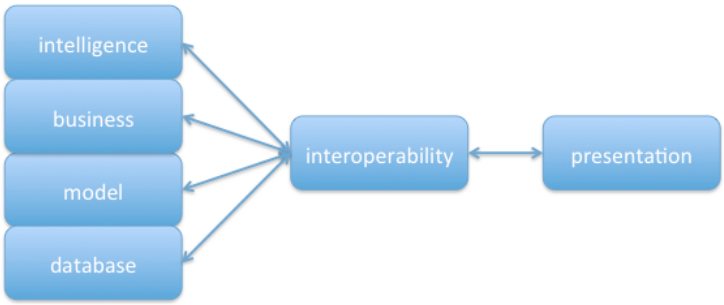

Figura 26 - Camadas do GeoHS

O pacote *business* é o repositório de classes relacionadas às regras de negócio da aplicação. As regras ao salvar um determinado foco de doença são implementadas nesse pacote. Já o pacote *database* resolve a comunicação com a camada de dados. O pacote *exception* trata as exceções específicas do projeto, como, por exemplo, *UserNotExistException* (lançada no momento que a aplicação detecta que o usuário não existe no cadastro) e *XmlProcessException* (exceção

lançada quando ocorre um erro no processo de importação e conversão do XML do CID-10 para a entidade de doenças).

O pacote *intelligence* é responsável por armazenar regras para serem consumidas no momento da inferência. As classes contidas nesse pacote fazem uso das bibliotecas do Jena para representação das triplas em RDF, criação de ontologias e regras de inferência. Logo abaixo desse pacote, está o *interoperability*, onde ficam armazenadas classes necessárias para converter XML em objetos e o local dos serviços REST que estão disponíveis no GeoHS, como é o caso do *upload* mostrado no quadro 01.

A linha 01 do quadro 01 indica o caminho relativo para se chegar ao serviço (também chamado de recurso) *doencas*. O método *upload* na linha 10 é anotado com '@Path("/upload")', indicando que o recurso está no caminho relativo "/doencas/upload".

A auditoria do sistema deve ter acesso a arquivos de *log* para identificar o comportamento dos sistemas, além de verificar a causa de erros ocorridos nos sistemas. Existe a classe *LoggerFactory*, a qual fabrica os *loggers* necessários no GeoHS.

```
1.  @Path("/doencas")
2.  public class DoencaEndpoint {
3.  @Inject
4.  private DoencaService service;
5.  @Inject
6.  @XmlReader
7.  private XStream reader;
8.  @POST
9.  @Path("/upload")
10. @Consumes(MediaType.MULTIPART_FORM_DATA)
11. public Response upload(@MultipartForm FileUploadForm form) {
12. Cid10 cid = (Cid10) reader.fromXML(new String(form.getData()).replace("&", ""));
13. cid.getDoencas().parallelStream().forEach(service::salvar);
14. return Response.status(200).build();
15. }...
```

Quadro 01 - Webservices para realizar upload de XML

Para a criação de páginas web, foi utilizado o framework JSF, baseado em componentes. A versão do JSF é a 2.2, mais atual, escolhida por ser a versão onde há mais facilidade para o desenvolvimento, envolvendo pouco conhecimento específico de JSF ao se criar e editar um arquivo XHTML. Como se pode ver no quadro 01, abaixo, o HTML é desenvolvido normalmente. Por exemplo, quando se deseja utilizar um componente JSF, basta adicionar o prefixo "jsf:" na propriedade adequada.

```
<form jsf:id="loginForm" class="form account-form">
        <div class="form-group">
            ...
            <input jsf:id="username" jsf:value="#{loginController.usuario.login}" type="text"
class="form-control" placeholder="#{i18n['login.form.placeholder.username']}" jsf:tabindex="1"/>
        </div>
</form>
```

Quadro 02 - Utilização de *tags* JSF

A estrutura da camada de visão contém o diretório "doencas", que possui as páginas responsáveis por listar, incluir e importar doenças; o diretório "error" que possui as páginas de erro para tratar as falhas do sistema; a pasta de usuários para conter as páginas de cadastro de usuário; a pasta "resources", responsável por manter arquivos de estilo, fonte e dinâmica das páginas, bem como mídias em geral que servem de recursos para a aplicação; a pasta "WEB-INF" que contém a estrutura e as configurações dos *Contexts and Dependency Injection* (CDI) e JSF, além da página de *login* e arquivos de configuração de segurança e do Wildfly. A figura 27 mostra esse modelo.

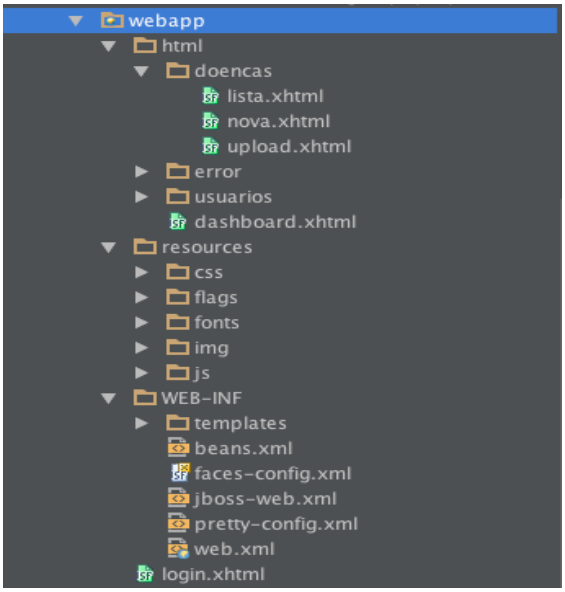

Figura 27 – Camada Web do GeoHS

No GeoHS, o CDI é utilizado tanto para o gerenciamento de transações, quanto para a injeção de dependência. A classe *DoencaService*, no quadro 03, demonstra estes dois tipos de funções do CDI. Na linha 01, o "@Transactional" é uma anotação de classe, indicando que todos os métodos da classe *DoencaService* são transacionais, ou seja, são executados como uma transação. Assim, os dados só serão armazenados no banco de dados se todos os passos dos métodos forem executados com sucesso. Caso contrário, os dados não serão salvos.

Na linha 04, o termo "@Inject" sinaliza que a variável *logger* da linha 05 é injetada (criada) a partir do CDI. Ou seja, o desenvolvedor não precisa se preocupar em criar uma nova instância da classe *Logger*, pois o CDI é responsável por isso. A variável *logger* tem a função de registrar os *logs* do sistema. Normalmente, os *logs* são instalados nos pontos de código onde ocorrem exceção, para que, numa auditoria, seja fácil identificar o erro e os passos que acarretaram o erro.

Nas linhas 07 e 08, é criado um *entityManager*, gerenciador de entidade do JPA, que representa uma interface para executar métodos de acesso ao banco de dados. Na linha 09, por exemplo, o objeto *entityManager* é utilizado para persistir uma doença na base de dados.

```
1. @Transactional
2. public class DoencaService implements Serializable {
3. ...
```

```
4.  @Inject
5.  private Logger logger;
6.  @Inject
7.  private EntityManager entityManager;
8.  public Doenca salvar(@Valid Doenca doenca) {
9.  entityManager.persist(doenca);
10. return doenca;
11. }
12. ...
```

Quadro 03 - Classe *DoencaService* mostrando funcionalidades do CDI no COISA

4.2.2. Telas do GeoHS

O login do sistema GeoHS tem dois campos de entrada (usuário e senha) e o botão para validar os campos e entrar no sistema. A tela de login é mostrada na figura 28 a seguir:

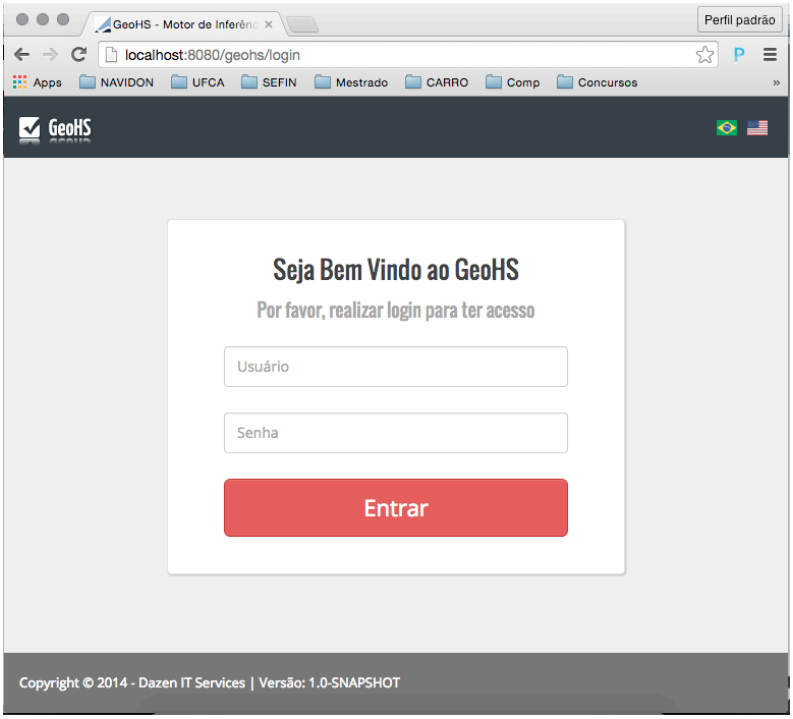

Figura 28 - Tela de login do GeoHS

A seguir, a figura 29 ilustra a tela de listagem de doenças. Nessa lista contém as doenças cadastradas com a importação do arquivo CID-10 ou doenças que o administrador precise criar.

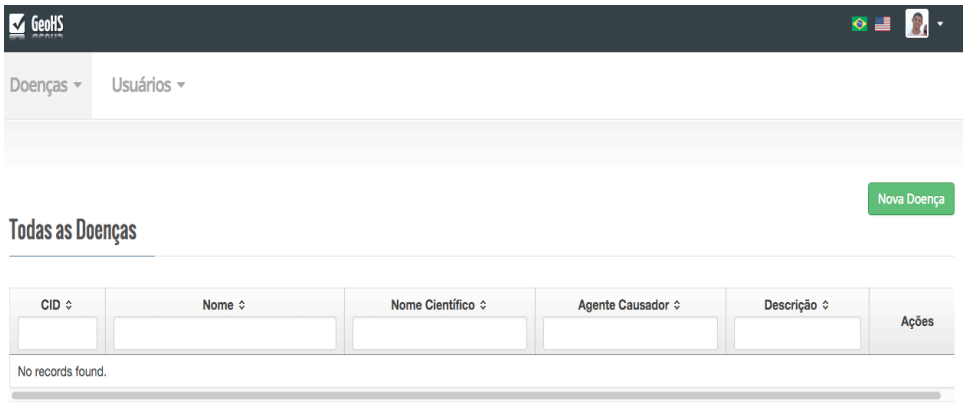

Figura 29 - Listagem de doenças

Ao clicar em "Nova Doença", o usuário terá acesso a tela de cadastro de doença, mostrada na figura 30 a seguir:

Figura 30 - Cadastro de doença

De posse do arquivo XML do CID-10, o usuário administrador pode realizar a importação de doenças pela tela mostrada a seguir:

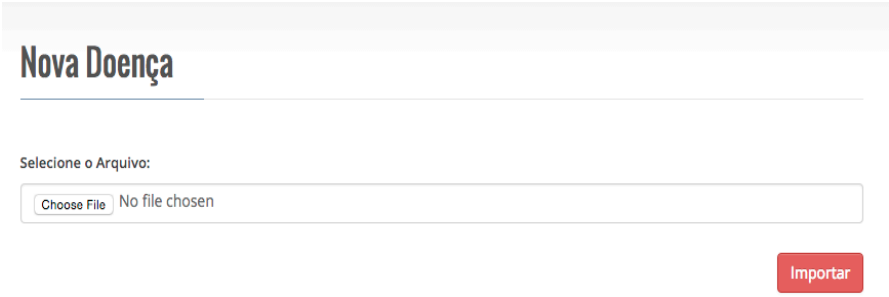

Figura 31 - Importar doenças do CID10

No sistema também é possível realizar o cadastro de usuários e associar as permissões condizentes com o papel de cada usuário. As figuras 32 e 33 mostram as listagens de usuários e permissões.

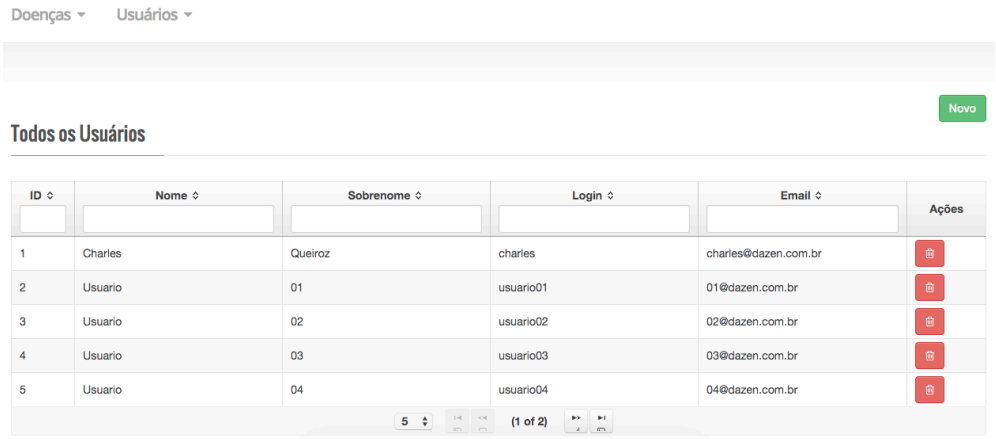

Figura 32 - Listagem de usuários

Figura 33 - Listagem de permissões

4.2.3. Testes

Os GeoHS possui testes unitários com a utilização do *JUnit* e de um *framework* de testes chamado *Arquillian*, capaz de simular o servidor de aplicação *Wildfly* no momento de injetar as dependências gerenciadas pelo CDI.

4.3. COBILE

O COBILE é o cliente móvel responsável pelas seguintes funcionalidades:

- Escolher destino e expor os riscos: Assim que o usuário escolher o destino, o aplicativo deverá retornar se há risco para o usuário se deslocar para o local desejado;
- Notificação de risco enquanto se desloca: Ao se movimentar, o aplicativo verifica se o usuário se aproxima de uma área de risco. Se for detectada alguma área de risco, o usuário é notificado.

4.3.1. Arquitetura do COBILE

A arquitetura do COBILE segue a estrutura sugerida pela Google, implementada pela IDE Android Studio, como mostra a figura 34.

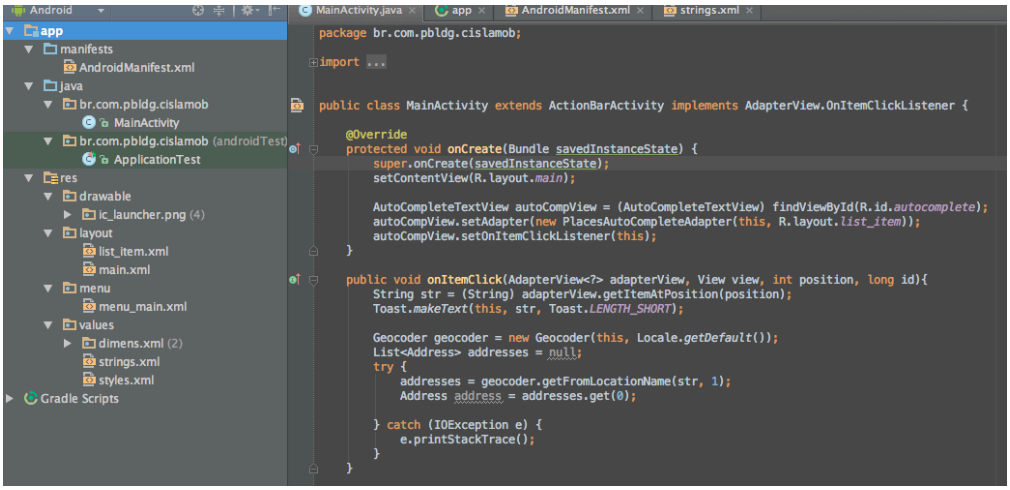

Figura 34 - Projeto COBILE no Android Studio

4.3.2. Telas do sistema

O COBILE inicia com uma tela onde o usuário pode escolher entre *escolha de destino* e *áreas de risco*. A figura a seguir apresenta este menu.

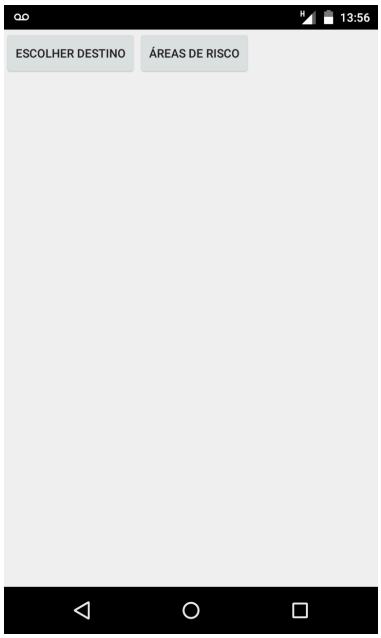

Figura 35 – Tela inicial do COBILE

As figuras 36 e 37 ilustram a *escolha de destino* do COBILE, onde o aplicativo sugere as cidades de acordo com o que o usuário digita no momento. Foi utilizada a API do Google para realizar a sugestão de cidades no momento da digitação.

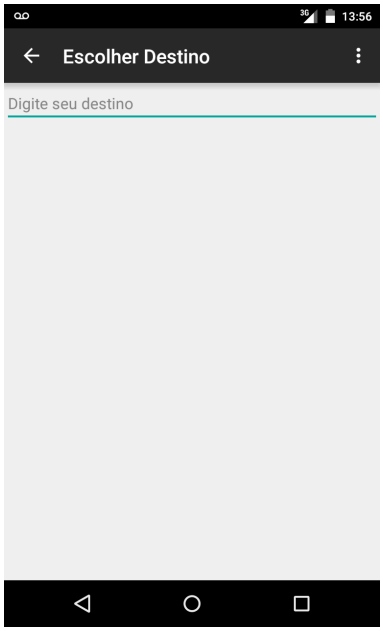

Figura 36 - Escolha de destino no COBILE

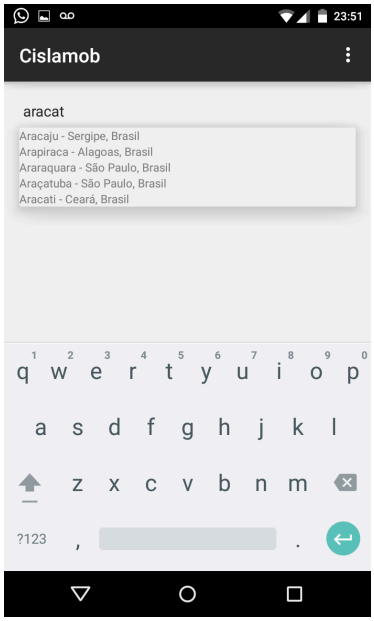

Figura 37 – Sugestões de escolha de destino no COBILE

Assim que o usuário seleciona o destino, o aplicativo retorna uma mensagem explicando ao usuário se existe ou não risco à sua saúde. A figura 38 apresenta esse resultado.

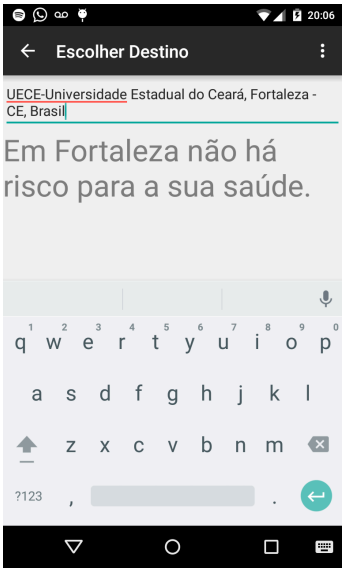

Figura 38 – Resultado sobre risco à saúde do usuário após escolha de destino

Ao escolher a opção *Áreas de risco*, o usuário tem a possibilidade de adicionar e remover áreas de risco, o que significa ativar ou desativar as áreas de risco para o usuário. É preciso o usuário ter o GPS ativado no dispositivo móvel para que a funcionalidade *Áreas de risco* trabalhe como o esperado. A figura 39 representa o cenário em que o usuário escolheu adicionar áreas de risco.

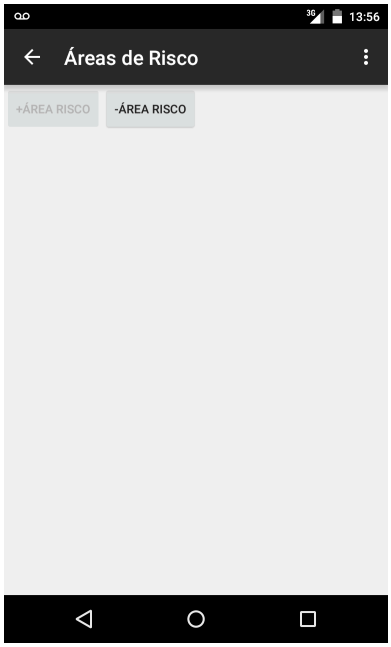

Figura 39 – Adicionar área de risco para o usuário

Após adicionar áreas de risco, o usuário pode ser notificado, recebendo alertas no dispositivo sobre ocorrências de doenças, baseados na localização e no

prontuário eletrônico do usuário. As figuras 40 e 41 mostram alertas ao usuário quando entra ou sai de uma região de risco.

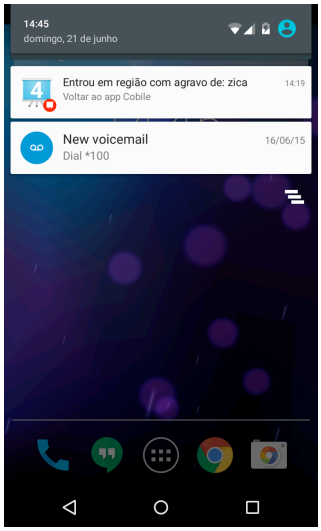

Figura 40 – Usuário recebe notificação ao entrar em uma área de risco

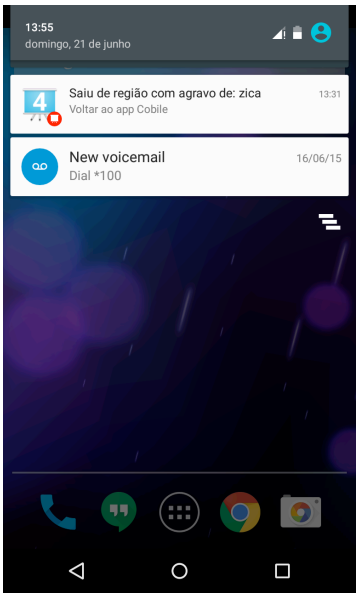

Figura 41 – Usuário recebe notificação ao sair de uma área de risco

5. CONCLUSÃO

A inclusão de um módulo do LARIISA capaz de monitorar e alertar sobre doenças foi o principal objetivo deste trabalho. Os cenários apresentados foram de usuários capazes de receber alertas sobre riscos de saúde. Para cada usuário, há um conjunto de ameaças possíveis baseado nos seus antecedentes, representados por registros eletrônicos de saúde, e na sua localização.

De início foi realizada uma análise sobre os sistemas de saúde existentes e como melhorar o panorama de saúde por meio da plataforma LARIISA. Detectou-se, então, a dificuldade que um cidadão tem pesquisar e receber informações sobre seu próprio estado epidemiológico, bem como a dificuldade da gestão pública em integrar sistemas de monitoramento de agravos de doenças com sistemas que possam realizar análise inteligente de dados.

A forma atual adotada pelo governo brasileiro para conter dados sobre ocorrências de doenças em uma determinada região é baseada no SINAN. Como visto anteriormente, o SINAN não oferece uma solução ideal para extração de seus dados, dificultando também a governança da área de saúde para coleta de relatórios para tomadas de decisão.

Na perspectiva de histórico dos pacientes, foi analisado que o governo, no presente momento, não mantém um modelo para integração desses dados com outros sistemas de saúde além do SUS. O impacto disso é que, ao se consultar pela primeira vez em um hospital de uma instituição privada, o paciente tem seus dados salvos apenas nessa instituição. Ou seja, se o mesmo paciente vai para um outro hospital de origem privada ou até para um posto público de saúde, os dados terão que ser inseridos novamente, não havendo integração entre os hospitais.

Este trabalho considerou todos esses problemas citados anteriormente e propôs um sistema, como um módulo do LARIISA, para minimizar ou anular tais situações desfavoráveis para os sistemas de saúde.

Como contribuição, foi adicionado um domínio de governança ao LARIISA que trata o monitoramento e os alertas de sobre doenças. Este domínio define regras, com auxílio do padrão internacional para modelo clínico, o *openEHR,* que antecipam agravos de doenças endêmicas e sinaliza usuários e a governança da área de saúde pública sobre possíveis endemias.

Além do domínio que envolve o monitoramento de doenças aos pacientes, o LARIISA ganha um protótipo que mostra, como uma prova de conceito, o funcionamento do COBILE, sistema que emite alertas aos usuários dependendo de um dos eventos: entrar ou sair de uma região de risco.

O trabalho desta dissertação também apresentou uma intensa pesquisa nas áreas de contexto, localização, ontologia e saúde, necessárias para a produção do sistema COISA, colaborando efetivamente no referencial teórico envolvido no LARIISA.

Por fim, este trabalho também agrega valor ao desenvolvimento de software dos sistemas do LARIISA, pois utilizou conceitos de gerência de configuração, o que torna o projeto mais fácil de ser continuado por uma equipe de desenvolvedores.

Para a continuidade dos projetos, é proposto o seguinte conjunto de trabalhos futuros:

- Criar módulo de redes sociais para detectar as palavras relacionadas a saúde que são mais utilizadas e a partir disso emitir relatórios para a governança de saúde;
- Continuar o desenvolvimento do COISA para cobrir todas as funcionalidades descritas nos requisitos funcionais apresentadas no capítulo 03;
- Regras inseridas por médicos: até o momento a aplicação é capaz de cadastrar regras e inferir regras a partir de uma situação baseada em contexto. É necessário analisar, juntamente com especialistas em saúde, um conjunto de regras específicas para doenças cadastradas, afim de o sistema poder inferir sobre determinado perfil de usuário e tomar a decisão correta. Por exemplo, um especialista pode especificar a regra que define que se o usuário nunca teve catapora e está próximo de agravos de catapora, este usuário deverá ser alertado.
- Sistema capaz de alertar unidades administrativas do governo em caso de alerta endêmico grave, como o vírus ebola, por exemplo. Ao ser diagnosticada uma doença, e a doença for considerada grave, é necessário alertar a governança da administração pública para que sejam tomadas as devidas providências. Esta notificação à governança

é feita de maneira automática e sensível ao contexto. Assim que o sistema perceber que há doença grave, emite o alerta.

- Utilizar a API do Google específica para comandos de voz, abrindo possibilidades de desenvolvimento de funcionalidades voltadas para o público deficiente visual. Esta API permite usuários falarem algo e o dispositivo móvel entende e realiza a ação necessária.

REFERÊNCIAS

AGILE, AGILE ALLIANCE. Agile development principles and pratices. Disponível em http://www.agilealliance.org. Acesso em: 3 jan. 2015.

ANDROID, Google. **Android, the world's most popular mobile platform.** Disponível em <http://developer.android.com/about/index.html>. Acesso em: 15 de Novembro de 2014.

BERNERS-LEE, TIM. Web for real people. 2005. Disponível em <http://www.w3.org/2005/Talks/0511-keynote-tbl/>. Acesso em: 7 out. 2014.

BRASIL. Ministério da Saúde. **Conceitos e definições em saúde**. Esplanada dos Ministérios - Bloco 11 - 8º andar. 70.000 - Brasília – Brasil. 1977

BRASIL, Ministério da Saúde. Secretaria Executiva. **Controle de Endemias**. Brasília – DF: Ministério da Saúde, 2001. 36 p. Disponível em <http://www.fef.com.br/biblioteca/arquivos/data/endemias.pdf>. 2001

BRASIL. Ministério da saúde. DATASUS. **Classificação Estatística Internacional de Doenças e Problemas Relacionados à Saúde – CID-10**. 2008. Disponível em <http://www.datasus.gov.br/cid10/V2008/cid10.htm>. Acesso em: 12 nov. 2014.

BRASIL. Ministério da Saúde. **Portaria n. 104**, de 25 de Janeiro de 2011.

BRASIL. Ministério da Saúde. Portal da Saúde. **Sistema Único de Saúde**. Disponível em <http://portalsaude.saude.gov.br/index.php/cidadao/entenda-o-sus>. Acesso em: 7 out. 2014.

BRASIL. Ministério da Saúde. **Programa Saúde da Família**. Disponível em <http://www.saudedafamilia.org/projetos/psf/psf.htm>. Acesso em: 7 out. 2014.

BRASIL. Ministério da Saúde. Portal da Saúde. **Medidas de prevenção e controle**. 22 abr. 2014. Disponível em <http://portalsaude.saude.gov.br/index.php/o-ministerio/principal/leia-mais-o-ministerio/664-secretaria-svs/vigilancia-de-a-a-z/malaria/l2-malaria/12194-como-se-prevenir>. Acesso em: 25 mai. 2015

BRASIL. Ministério da saúde. **Portaria n. 1.271**, de 06 de Junho de 2014.

BVSMS, Biblioteca Virtual em Saúde do Ministério da Saúde. **Registro Eletrônico de Saúde**. Disponível em <http://bvsms.saude.gov.br/bvs/IIIfis/pdf/Rogerio_Sugai.pdf>. Acesso em: 03 jan. 2015.

CKM, openEHR Foundation. ***Clinical Models Program***. Disponível em <http://www.openehr.org/programs/clinicalmodels/>. Acesso em: 15 de Fevereiro de 2015.

CONASS, Conselho Nacional de Secretários de Saúde. **Situação Atual Do Sistema de Informações de Agravos de Notificação - SINAN**. 2013. Disponível em <http://www.conass.org.br/NT%2045-%202013%20SINAN%20.pdf>. Acesso em: 17 de Fevereiro de 2015.

DEY, A. K. **Providing Architectural Support for Building Context-Aware Applications**. In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science. College of Computing Georgia Institute of Technology, 1999

DROOLS, JBoss Comunity. **Overview**. Disponível em <http://www.drools.org/>. Acesso em: 17 de Fevereiro de 2015.

GARDINI, L. M.; BRAGA, R.; BRINGEL, J.; OLIVEIRA, C.; ANDRADE, R.; MARTIN, H.; ODORICO, L.; ANDRADE, M.; OLIVEIRA, M. **Clariisa, a Context-Aware Framework Based on Geolocation for a Health Care Governance System**, Instituto Federal de Educação, Ciência e Tecnologia – IFCE, 2013

GEOFENCE, Google Android API, **Creating and Monitoring Geofences**. Disponível em <https://developer.android.com/training/location/geofencing.html>. Acesso em: 8 de Outubro de 2014.

GIT, git-scm. ***git***. Disponível em <http://git-scm.com/>. Acesso em: 02 de Fevereiro de 2015.

GLEBA, K.; ŚLIWA, J.; DUDA, D.; GLOWACKA, J.; PYDA, P. ***Run-time ontology on the basis of event notification service***. *Communications and Information Systems Conference* (MCC). *Military Communication Institute*, 2012

GRAPE STUDIOS. **Geofence Tracker Tasker free**. 28 jul. 2014. Disponível em <https://play.google.com/store/apps/details?id=sh.geofence.tracker.tasker>. Acesso em: 4 nov. 2014.

GUARINO, N.; GIARETTA, P. **Ontologies and knowledge bases: towards a terminological clarification**. In: MARS, N. (Ed.). Towards very large knowledge bases: knowledge building and knowledge sharing. Amsterdan: IOS Press, 1995. p. 25–32. Disponível em: 140

GUARINO, N. **Formal Ontology in Information Systems**. Amsterdan: IOS Press, 1998. 341 p.

GUERMAH, H.; FISSAA, T.; HAFIDDI, H.; NASSAR, M.; KRIOUILE, A. **Ontology based Context Aware e-Learning System**. IMS Team, SIME Lab, ENSIAS, Mohammed V Souissi University. 2013

ISO/TR 20514:2005. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION – ISO. **Health informatics, Electronic health record, Definition, scope and context**. 2005. Disponível em: <https://www.iso.org/obp/ui/ - iso:std:iso:tr:20514:ed-1:v1:en>. Acesso em: 25 mai. 2015.

ISO 13606. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/TC251 13606 Health informatics - Electronic record communication - Part 1: Reference Model and Part 2: Archetype interchange.** 2008.

JENKINS, Jenkins. **Meet Jenkins**. Disponível em <http://jenkins-ci.org/>.Acesso em: 15 de Novembro de 2014.

JIRA, Atlassian. **Plan, track, work - smarter and faster**. Disponível em <https://www.atlassian.com/software/jira>. Acesso em: 02 de Fevereiro de 2015.

JUNITANDROID, Google. Testing Fundamentals. Disponível em <http://developer.android.com/tools/testing/testing_android.html>. Acesso em 15 de Novembro de 2014.

KO, K.; SIM, k. **Development of Context Aware System based on Bayesian Network driven Context Reasoning Method and Ontology Context Modeling**. School of Electrical and Electronics Engineering, Chung-Ang University, Seoul 156-756, Korea. 2008

LIU, WEI; LI, XUE; HUANG, DAOLI. **A Survey on Context Awareness**. School of Automation Beijing University of Posts and Telecommunications. 2011

MCGUINNESS, D. L.; HARMELEN, F.V. **OWL Web Ontology Language.** World Wide Web Consortium. Knowledge Systems Laboratory, Stanford University. Vrije Universiteit, Amsterdam. 2004

MENEZES, J. V.; D'CASTRO, R. J.; RODRIGUES, F. M. M.; GUSMÃO, C. M. G.; LYRA, N. R. S.; SARINHO, S. W. **InteliMed: uma experiência de desenvolvimento de sistema móvel de suporte ao diagnóstico médico**. Grupo de Pesquisa S@BER: Tecnologias Educacionais e Sociais. Programa de Pós-graduação em Engenharia Biomédica, Centro de Tecnologia e Geociências (CTG). Universidade Federal de Pernambuco, Recife, Brasil. 2011

MENEZES, J.; GUSMÃO, C. **InteliMED – Proposta de Sistema de Apoio ao Diagnóstico Médico para Dispositivos Móveis**. Grupo de Pesquisa S@BER: Tecnologias Educacionais e Sociais. Programa de Pós-graduação em Engenharia Biomédica, Centro de Tecnologia e Geociências (CTG), Universidade Federal de Pernambuco, Recife, Brasil. 2013

MOBILETIME, **Mobile Time Tracking Tips**. Disponível em <http://www.mjobtime.com/blog/bid/28647/Geo-fencing-For-mJobTime-s-Mobile-Time-Tracking-Software>. Acesso em: 04 nov. 2014.

NOY, N. F.; MCGUINNESS, D. L. **Ontology Development 101: A Guide to Creating Your First Ontology**. Stanford University, Stanford, CA, 94305. Disponível em <http://protege.stanford.edu/publications/ontology_development/ontology101.pdf>. Acesso em: 22 out. 2014.

OLIVEIRA, M; HAIRON, C.; ANDRADE, O.; MOURA, R.; SICOTTE, C.; DENIS, J-L.; FERNANDES, S.; GENSEL, J.; BRINGEL, J.; MARTIN, H. **A context-aware framework for health care governance decision-making systems: A model based on the Brazilian Digital TV**. In: 2010 IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS MOBILE AND MULTIMEDIA NETWORKS, 2010, Montreal. Anais Montreal, 2010.

OPENEHR, openEHR Foundation. ***What is openEHR*?** 2015. Disponível em <http://www.openehr.org/what_is_openehr>. Acesso em: 15 fev. 2015

OPENEHR, openEHR Foundation. **What parts of openEHR have Ontological Relevance?** 2015. Disponível em: <https://openehr.atlassian.net/wiki/display/ontol/Ontologies+Home>. Acesso em: 21 fev. 2015

PROTÉGÉ, Stanford University. Stanford, CA. Disponível em <http://protege.stanford.edu>. Acesso em: 15 jul. 2014.

SANTOS, M. R.; BAX, M. P.; PESSANHA, C. **Uma Leitura Ontológica da Norma ISO 13606 para o Registro Eletrônico de Saúde**. Escola de Ciência da Informação – Universidade Federal de Minas Gerais (UFMG). 2010

SCHILIT, B.; ADAMS, N.; WANT, R. **Context-Aware Computing Applications**. In: FIRST WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, Santa Cruz, 1994. p. 85-90.

SCHILIT, B.; THEIMER, M.M. **Disseminating Active Map Information to Mobile Hosts**. IEEE Network Septembed/October 1994, pp.22-32.

SCRUM, Scrum.org. **What is Scrum? A better way of working**. Disponível em <https://www.scrum.org/Resources/What-is-Scrum>.Acesso em: 15 de Novembro de 2014.

SIMDA. **Sistema de Monitoramanto Diário de Agravos**. Disponível em <http://tc1.sms.fortaleza.ce.gov.br/simda/login/auth>. Acesso em: 13 de Maio 2014.

SINAN. **Sistema de Informação de Agravos de Notificação**. 2014. Disponível em <http://dtr2004.saude.gov.br/sinanweb/>. Acesso em: 13 mai. 2014.

SISA, Antunes, R. **SISA - Uma aplicação sensível ao contexto para agravos de Dengue: uma prova de conceito do projeto Lariisa**. Universidade Estadual do Ceará; Instituto Federal de Educação, Ciência e Tecnologia do Ceará. 2011

SONAR, SonarQube Software. **Documentation for SonarQube**. Disponível em <http://docs.codehaus.org/display/SONAR/Documentation>. Acessso em: 15 de Novembro de 2014.

STRANG, T.; LINNHOFF-POPIEN, C. **A Context Modeling Survey**. In: WORKSHOP O GRAPHICAL MODELS, 2004. Anais... 2004. p. 1-8.

SUBVERSION, Apache. **Apache Subversion Documentation**. Disponível em <https://subversion.apache.org/docs/>. Acesso em: 15 de Novembro de 2014.

TYAGI, S. ORACLE. RESTful Web Services. 2006. Disponível em http://www.oracle.com/technetwork/articles/javase/index-137171.html. Acesso em: 27 mai. 2015

USCHOLD, MIKE; GRUNINGER, MICHAEL. **Ontologies: Principles Methods and Applications**. Knowledge Engineering Review. Artificial Intelligence Applications Institute. 1996

XP, Extreme Programming. **Extreme Programing: A gentle introduction**. Disponível em <http://www.extremeprogramming.org/>. Acesso em: 15 de Novembro de 2014.